commodore 16 * commodore plus/4

BEVEZETÉS A BASIC-NYELVBE

1. rész



NOVOTRADE

COMMODORE 16 – COMMODORE PLUS/4

BEVEZETÉS A BASIC NYELVBE

1. rész

BUDAPEST, 1986

Fordította: KÍGYÓS ERZSÉBET
Átdolgozta: DR. LENGYEL JÓZSEF
Lektorálta: VARGA ANDRÁS
Felelős szerkesztő: TARR KÁLMÁNNÉ

A kiadásért felel: RÉNYI GÁBOR, a NOVOTRADE RT. igazgatója
Kiadványmenedzser: BÉKÉS TAMÁS
Fedélterv és tipográfia: DÉVÉNYI ERIKA
(16,5 A/5 ív)
ISBN 963 02 40351
Készült a Gutenberg Nyomdaipari és Reklámszervező Kisszövetkezet közreműködésével
VAJAPRINT

ELŐSZÓ

Ez a könyv a hozzá tartozó programokkal egy önképző tanfolyamsorozat I. része, amellyel a COMMODORE 16, ill. Plus/4 programozásának mind tökéletesebb elsajátításában szeretnénk segíteni a felhasználókat. Ebben megismerheti a programkészítés alapelveit és a BASIC programozási nyelv alapjait. A tanfolyam három fő részből áll:

1. Egy 15 leckéből álló „tanulj-magad" szöveg, amely a programozás egy-egy fontos lépésével ismertet meg.

2. A 2 db magnókazettán, ill. az 1 db lemezen olyan programgyűjtemény van, amely segít a leckék megtanulásában.

3. Bemutatjuk a folyamatábra tervezés alapjait, amelyet profi „számítástechnikusok" használnak. Az ilyen terv segítséget nyújt ahhoz, hogy Ön megbízható, hibátlan, hatékony és tömör programokat tervezzen.

Meg kell jegyeznünk, hogy ez a könyv megtanítja Önt hasznos és szórakoztató programok készítésére, de *nem* tanítja a BASIC nyelv egészét. A BASIC haladóbb részeit alaposan tárgyalja és magyarázza e sorozat második része.

MEGJEGYZÉSEK

A Tisztelt Olvasó olyan önképző tanfolyamot kezdett most el, amelynek mindkét része angol forrásanyagok felhasználásával készült: mind a könyv, mind a hozzá tartozó programok (kazettán vagy hajlékony lemezen). Ennek kapcsán a következőkre szeretnénk felhívni figyelmét.

1. A programok mind szövegesek. Ezért az eredeti angol programokat a magyar nyelvű könyvvel nem lehet használni. A könyvhöz tartozó magyar nyelvű kazettákat a Novotrade RT. forgalmazza.

2. A programokat a magyar ékezetes betűkkel ellátott, a Tudományszervezési és Informatikai Intézet által oktatási intézmények részére forgalmazott Commodore 16 gépekre írtuk át. Ha nem ilyen géppel rendelkezik, kissé nehezebb lesz a képernyőre írt szövegek elolvasása (minden ékezetes betű helyett valamilyen grafikus jel fog megjelenni). Célszerű, ha az ékezetes karaktereket előállító szoftvert beszerzi vagy a gépnek (Commodore 16, 116, Plus/4) karakter-PROM cseréjét, valamint billentyűzetének feliratozását elvégezteti a Novotrade RT-nél.

3. Az ékezetes betűk használata miatt kénytelenek voltunk a programokban a gépnek az ún. kisbetű/nagybetű vagy írógép üzemmódját alkalmazni, amelyet viszont a tanfolyam (I. része) nem tárgyal. Az ezzel kapcsolatos ismereteket megszerezheti a Commodore 16 Felhasználói kézikönyvből (NOVOTRADE kiadás, 1985). E könyvben – ahol szükséges – megjegyzésekkel utaltunk a problémára. Itt arra hívjuk fel a figyelmét, hogy a mellékelt programok listázását ugyancsak írógép módban célszerű végezni, és ekkor – a könyvben közöltektől eltérően – minden BASIC kulcsszó, függvénynév és változónév kisbetűvel jelenik meg, ill. ha változtatunk a programon (van ilyen feladat), akkor kisbetűvel kell azokat beírni. Ez bonyolultnak tűnik, azonban nagyon egyszerű: akár nagybetű/grafika, akár írógép üzemmódban vagyunk, a kulcsszavakat stb. mindig azonos módon kell begépelni: a SHIFT billentyű lenyomása nélkül.

Ha az e pontban leírtakat nem értette, olvassa el újra, ahányszor elérkezik a tanfolyam során egy fentiekkel kapcsolatos megjegyzéshez.

A lektor

TARTALOMJEGYZÉK

BEVEZETÉS

Üdvözöljük a COMMODORE 16 és PLUS/4 programozói „tanfolyamán”. Mindkét gép kiválóan alkalmas játékra, ragyogó és izgalmas képek és hangok létrehozására, de ugyanakkor önmagában teljes, modern számítógép is. A számítógépek hallatlanul sokoldalúak, többre képesek, mint bármi más – az ember kivételével. Számítógép lehet például oktatógép, kalkulátor, mozgássérültek segítőtársa, szövegszerkesztő, könyvelő- és készletgazdálkodó gép, intenzív osztályon beteg monitora, ipari folyamatvezérlő vagy mérnökök által alkalmazott tudományos számítógép, amellyel épületeket, erőműveket és repülőgépeket terveznek.

A számítógépek, s az általuk vezérelt rendszerek egyre nagyobb teret hódítanak mindennapi életünkben. Már ma is sok eszköz mögött – a közlekedési lámpák, a pénztárgépek és bankok termináljai stb. – ott van a számítógép. Ez a folyamat egész életünkben csak fejlődni fog. A világ most éli át a számítógépes forradalmat, amely éppoly következményeket hoz magával, mint annak idején az ipari forradalom.

A számítógépes forradalmat nem lehet megállítani, de – ha akarjuk – mindnyájan befolyásolhatjuk. A világ lassan kétféle embertípusra osztható: utasokra és pilótákra. Az utasok hagyják, hogy a dolgok megtörténjenek; lehet, hogy élvezik a számítógépek nyújtotta előnyöket, lehet, hogy gyűlölik, de az is lehet, hogy élvezik is, gyűlölik is. Gyakran hangoztatják a véleményüket, de ennek semmi hatása nincs – nem jutnak el az irányító szervezéshez, s ha mégis eljutnának, nem tudják, hogyan használják azokat.

A pilóták azonban kézben tartják az egész forradalmat. Új számítógéptípusokat találnak fel, és kigondolják, hogyan lehet hasznosan dolgozni velük. Nagy a pilóták felelőssége, hiszen rájuk hárul, hogy a világot a béke, a szabadság és a bőség felé navigálják, minél távolabb attól a rémálombeli társadalomtól, amelyet a sci-fi könyvek gyakran ábrázolnak.

Mi különbözteti meg a pilótát az utastól? Egyetlen dolog: annak a megértése, hogy hogyan működik a számítógép. Természetesen a megértésnek különböző szintjei vannak. A legtöbb ember tudja, hogyan kell működtetni egy játékautomatát, még ha nem is tudja elmagyarázni a mechanizmusát. (Igen, abban is számítógép van.) Az a szint, amire én gondolok, sokkal elmélyültebb. Annyira alapos és teljes, hogy annak birtokában bármit meg tudunk csináltatni a számítógéppel, amit csak akarunk: végezhet oktató tevékenységet, ipari vagy orvosi alkalmazást vagy pusztán szórakoztató játékokat.

Ahhoz, hogy ilyen hatalmunk legyen a számítógép felett, hogy a gép gyors, pontos és szolgálatkész szolgánk legyen, tudnunk kell *programozni* a gépet. Ez a feltétele annak, hogy pilótává váljunk.

Ez az egész könyv a programozásról szól. A COMMODORE 16-ra és PLUS/4-re vonatkozik, de ha az ember megtanulta ezeknek a gépeknek a programozását, könnyen át tudja alakítani a tudását bármilyen más számítógépre, kicsire vagy nagyra. Minél több programot készítünk, annál könnyebbé válik a feladat. Majd' minden ember meg tudja tanulni, hogyan kell programozni, ha veszi a fáradságot, és ezt Ön is megteheti. Nem kell túl sokat tudnia a matematikáról, de hasznos lenne, ha találna a munkájához egy csendes helyet, ahol olvashat, gondolkodhat és használhatja a számítógépet; és a legjobb, ha sok időt biztosít magának a tanfolyam elvégzéséhez. Ne siessen!

A tanfolyam 15 leckéből áll. Minden lecke általában egy-két esti komoly munkát igényel. Majd' minden lecke a következő részekből áll: elméleti ismeretek, gyakorlati munka a gépen, programozás, végül pedig egy „ön-teszt”, amelyen lemérheti, mennyire sikerült elsajátítani a leckét. Minden lecke „gyakorlatokat” tartalmaz, ezeket pipálja ki, ahogy elvégezte azokat.

Amikor a leckében kérdéseket teszünk fel Önnek, általában helyet hagytunk arra, hogy beírja a válaszát. Használja ezeket: írjon puha ceruzával és legyen kéznél egy radír, hogy ki lehessen radírozni a válaszokat, ha átadja valakinek a könyvet. Ha olyan példányból kezd tanulni, amelyben már benne vannak a válaszok, radírozza ki azokat, mielőtt elkezdi használni.

A programozás olyan szoros szövésű anyag, amelyben a gondolatok közvetlenül egymásból következnek. A korábbi leckében előfordult problémát újra nem magyarázzuk el. Nem megy például semmire a 10. leckével, ha nem olvasta el és nem értette meg az *összes* leckét 1-től 9-ig. Ezért igazán fontos a sorrend betartása.

Amikor egy új lecke tanulásába kezd, először olvassa végig az egészet. Nem marad meg sok a részletekből, de képet tud alkotni

arról a témáról, amelyet a következőkben meg fog tanulni.

Ezután dolgozza fel alaposan a leckét. Minden részlet számít, és az a rész a legfontosabb, amelyik a legnehezebbnek tűnik. Ne ugorjon át semmit, próbáljon mindent megérteni! Amikor úgy érzi, hogy megtanult valamit, mondja el magának a saját szavaival. És ne guruljon dühbe, ha többször újra kell olvasnia a lecke bizonyos részleteit, sőt, hogy vissza kell lapoznia egy korábbi leckéhez, hogy fényt derítsen egy homályos pontra. Ez teljesen szokványos műszaki témáknál.

A programozás olyan, mintha hangszeren játszana az ember: gyakorlással lehet megtanulni. Ezért meg kell oldania a könyvben szereplő, minden programozással kapcsolatos feladatot. Amint képesnek érzi magát rá, kezdje el feltárni és megoldani a saját problémáit.

Ha befejezte ezt a könyvet, több különböző célra tudja majd használni a számítógépet. Például megszerkeszthet vele teszteket vagy eljátszathatja vele az Ön által kitalált játékokat, de számlák, pénzösszegek rendszerezésében is hasznosnak tartja majd. A játékok vagy egyéb alkalmazások tartalmazhatnak Ön által tervezett színes mozgó képeket, vagy az Ön elképzelése szerinti hangokat, gyönyörű dallamokat vagy durva zörejt!

Mindazonáltal a programozás hatalmas téma, senki sem tudná felölelni az egészet egyetlen „tanfolyamon" belül. Egy idő múlva bizonyára Ön is tovább szeretne majd lépni. Esetleg érdeklődni fog komplikáltabb problémák megoldása iránt, vagy használni akarja a gépet egy kisvasút vagy egy telefonközpont irányítására. Ennek a sorozatnak a második kötete: Bevezetés a BASIC nyelvbe – II. rész, segíteni fog Önnek abban, hogy meg tudja oldani ezeket az összetett és igényes feladatokat.

No, elég a szóból. Itt az ideje elkezdeni az 1. leckét. Sok szerencsét!

1. LECKE

1.1 GYAKORLAT

1.2 GYAKORLAT

1.3 GYAKORLAT

Ez a könyv két Commodore típusú számítógépről szól. Az egyik a Commodore 16, a másik a bonyolultabb Plus/4. A két gépnek más-más a külseje, különböző méretű tárolóik vannak, és más elrendezésű a billentyűzetük is, de a programozás fontosabb kérdéseiben azonosak. Egy könyv elegendő mind a kettőhöz.

Ahol a különbségek számottevőek (főleg az első leckékben), külön-külön írjuk le a gépeket. A részeket kerettel különböztetjük meg.

Az első bekeretezett rész mindig a Commodore 16-ra, a második mindig a Plus/4-re vonatkozik így:

Ez a Commodore 16-ra vonatkozik (és nem érvényes a Plus/4-re).

Az itt leírtak csak a Plus/4-re vonatkoznak, és semmi közük a Commodore 16-hoz.

Az 1. leckében megmutatjuk, hogyan kell üzembe helyezni a számítógépet. Megmagyarázunk számos általános tudnivalót és bemutatjuk azokat a gyakorlati problémákat, amelyek gondokat okozhatnak annak, aki először vásárol számítógépet.

Ahhoz, hogy megtanuljunk programozni, megfelelő környezetre van szükségünk. Keressünk egy csendes, kényelmes helyet, és hosszabb időre (legalább 2 órára) tervezzük a foglalkozást, és akkor, amikor nem vagyunk túl fáradtak ahhoz, hogy koncentráljunk.

Próbáljunk minden zavaró körülményt elhárítani – tegyünk figyelmeztető cédulát az ajtóra, húzzuk ki a telefont és közöljük az egész családdal, hogy nem érünk rá; semmi nem nehezíti meg annyira a programozást, mint az, ha állandóan félbeszakítják az embert.

Ha már üzembe helyezte és használta a számítógépét, kezdheti rögtön az első gyakorlatot. Ha nem, olvassa át gyorsan a leckét még akkor is, ha körülbelül tudja, miről szól; még mindig hasznosnak találhatja.

Először is helyezze el a számítógépet, a kazettás egységet vagy a lemezmeghajtót maga elé egy asztalra, a tápegységet leteheti a földre az asztal alá, és az egységek összekapcsolása után csatlakoztassa a hálózatba. A tv legyen kb. 2 m-re, ha kis, hordozható készülék; a nagyobb képernyőjű készülékeket még messzebbre kell állítani. Az elolvasandó szövegek és képek elég nagyok ahhoz, hogy a megszokott távolságból nézzük, de Ön is rájönne, hogy nagyon kényelmetlen és fárasztó úgy dolgozni, ha a képernyő túl közel van. A különböző egységeket úgy kapcsoljuk össze, ahogy az az ábrákon látható. A csatlakozókat enyhe, egyenletes nyomással kell a foglalatba helyezni. Soha ne erőltessük, de ellenőrizzük a csatlakozó elemeket az összekapcsolás előtt. A számítógépek nagyon strapabíróak, de a csatlakozó dugók és foglalatok könnyen megsérülnek és tönkremennek, ha túl sokszor kapcsolgatják össze és szét. Próbáljuk meg a lehető leghosszabb ideig összedugaszolt állapotban hagyni a gépet.

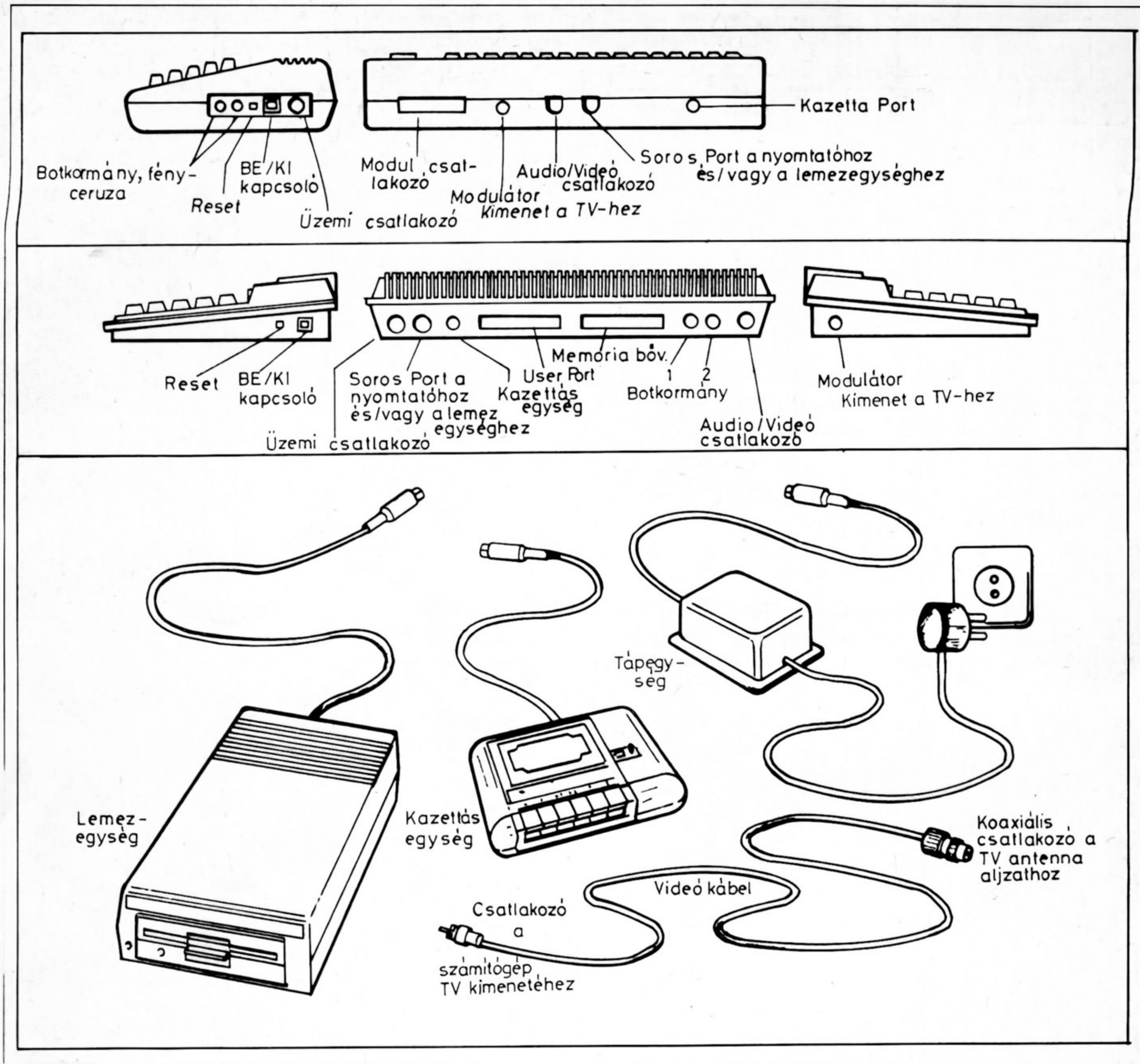
Ha a tv-jének kettős feladata van, és mint tv-készüléket is használja, szerezzen be egy antenna kapcsolót, amivel egyszerre lehet bedugaszolva tartani a számítógépet és a tv-antennát.

A számítógépet is, a tv-t is működtetheti közös hálózati hosszabbító vezetékkel. Minél hosszabb a vezeték, annál szabadabban helyezheti el a gépet. Ha kazettás egységet használ, elegendő egy két dugaszolóaljzatos hosszabbító, ha lemezegysége van, javasoljuk a négy dugaszolóaljzatost, ugyanis a tartalék hasznos lesz, ha egy nyomtatót is beszerez.

Ha úgy dönt, hogy hosszabbító kábelt használ, azt ajánljuk, ne Ön szerelje, hacsak nem ért igazán hozzá. Ha nem érti, miért javasoljuk ezt, akkor biztosan szüksége van valakinek a segítségére. Egy szakképzett villanyszerelő lesz a legjobb választás!

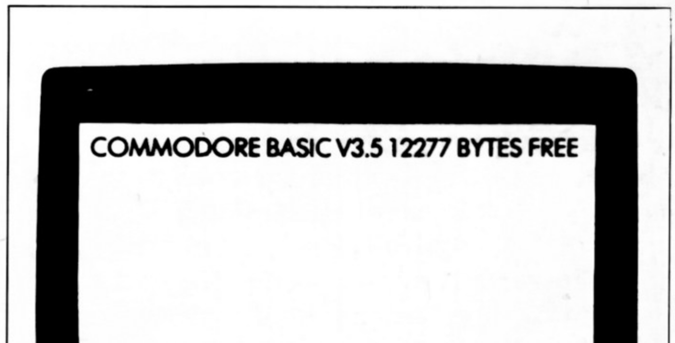
Most már bekapcsolhatja.

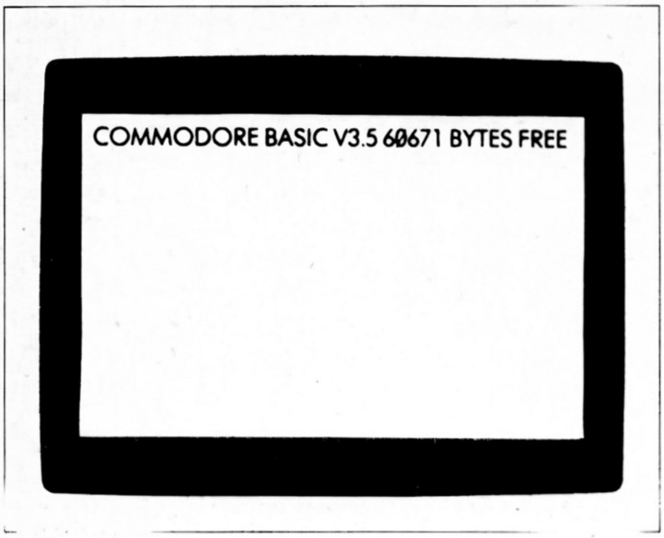
Kapcsolja be a tv-t. Ha egyszerű keresőgombja van, forgassa a 36. (UHF) csatornához vagy környékéhez. Ha nyomógombos, válasszon ki egy olyan csatornát, amelyet általában nem használnak vételre. A készülék képernyője üres lesz, és feltehetőleg hangosan sziszeg. Ha akarja, tekerje le a hangerőt.



Következő lépésként helyezze feszültség alá a számítógépet a jobb oldalon levő kapcsolóval. Ha minden rendben van, kigyullad a piros lámpa, de – hacsak nincs külön szerencséje – a tv-készülék még mindig nem mutat semmit.

Most menjen oda a tv-hez, és állítsa be. A beállítás pontos módja a gyártmánytól függ, de mindig szerepel a gyártó utasításai között. A legtöbb készüléknél valamilyen finom állító tartozik minden csatornához. Néha a beállítás kezelőgombjait elrejtik egy kis lap mögé. Ha csavarhúzót kell használnia, ne nyúljon be vele a készülék belsejébe, mert könnyen áramütés érheti. Ha a hangolás sikerült, a következő kép jelenik meg:





A képernyő fehér, türkizkék kerettel (Lehet, hogy állítani kell a sor- és képfrekvencián egy stabil kép eléréséhez.)

Ha nem jelenik meg ez a kép, kapcsolja ki a számítógépet néhány másodpercre, aztán próbálja újra.

Ha valami nincs rendben, ellenőrizze a következőket:

– Működik-e a tv? Próbáljon egy adást venni. (Ha szükséges, javíttassa meg.)

– Ég-e a számítógép piros lámpája? Ha nem, ellenőrizze:

a) hogy nincs-e általános hálózati feszültség-kimaradás;

b) hogy működik-e egy másik készülék (pl. asztali lámpa vagy hajszárító) ugyanabból a dugaszoló aljzatból;

c) hogy a számítógép tápegységében a biztosíték jó (próbálkozzon egy új biztosítékkal);

d) hogy a tápegység csatlakozója szorosan illeszkedik-e a számítógépbe.

– A számítógép jól van-e csatlakoztatva a tv-antenna bemenetéhez.

Ha a rendszer így sem működik (ez roppant valószínűtlen eset), javításra van szüksége.

A képernyőn megjelent üzenet számos „karakterből", többek között számokból és betűkből áll. Ezek a karakterek mindig azonos méretűek, és ha a monitor tele van, 1000 karakter fér el rajta. Az első sor azonosítja a terméket: ez egy BASIC rendszer, amelyet a Commodore Business Machines tervezett és kivitelezett. A V3.5 az aktuális BASIC rendszer verziószáma, amely mindannyiszor változik, valahányszor a gyártó a BASIC rendszert kisebb-nagyobb mértékben módosítja.

Végül pedig közli, hogy mennyi szabad tárhely (memória) van a gépében. Minden számítógépnek szüksége van egy „memóriára", ahol az elvégzett munka részleteit tárolja. A tárkapacitás mértékegysége a „byte", amely egy információ-szimbólumot vagy -karaktert tud hordozni. Minél nagyobb a gép memóriája, annál összetettebb feladattal tud megbirkózni. Az elérhető memóriahelyek száma Commodore 16 típusnál 12 277, a Plus/4 típusnál pedig 60 671. Ha kiegészítjük a számítógépet további memóriaegység csatlakoztatásával, ezek a számok növekedni fognak.

Ha kisebb szám van a képernyőn, mint várnánk, egyértelmű, hogy a számítógép rossz. Javíttatnia kell!

A következő sor azt jelzi, hogy a gép most kész (READY) engedelmeskedni azoknak a parancsoknak, amelyeket a billentyűzeten begépelünk.

A következő sorban egy villogó fekete négyzet van. Ez a kurzor. Begépeléskor a kurzor megmutatja a karakter pontos helyét a képernyőn. Például próbáljuk ki a következőt:

PRINT 5+8 RETURN

(ez 9 billentyű leütés:

PRINT5+8 és a RETURN gomb, amely a billentyűzet jobb oldalán van.)

Mielőtt elkezdene bármit beírni, érintse meg a SHIFT LOCK billentyűt, hogy meggyőződjön arról, hogy nincs-e lenyomott állapotban. Amint leírja az egyes jeleket, a RETURN -t kivéve mind megjelennek a képernyőn, a kurzor pedig mindig továbblép egy hellyel. A RETURN gomb fő feladata az, hogy a számítógéppel végrehajtasson egy parancsot, jelen esetben azt, hogy PRINT (azaz írja ki) 5 és 8 összeadásának eredményét.

A számítógépnek programmal kell rendelkeznie ahhoz, hogy bármilyen hasznos munkát tudjon végezni.

A programokat mágnesszalagok vagy hajlékony lemezek (floppy diskek) tárolják. Az első gyakorlatban azt gyakorolhatja, hogyan lehet betölteni egy programot a szalagról (kazettáról) vagy a lemezről a gépbe. De ha olyan szerencsés, hogy lemezegységgel rendelkezik, ugorjon rögtön az 1.2 gyakorlatra, amelyet speciálisan Önnek terveztek.

1.1 Gyakorlat

Programbetöltés kazettás egységről

1. Ellenőrizze, hogy a kazettás egység csatlakoztatva van-e a számítógéphez.

2. Nyomja meg az EJECT gombot a kazettás egységen.

3. Nyissa ki a kazettatartót és tegye be a tanfolyamhoz tartozó első kazettát. A felirata TAPE 1. Bármelyik oldala behelyezhető, mivel az oldalak egyformák. Mindenesetre a kazetta ablakának felénk kell néznie. Zárjuk be a kazettatartót. Ha nem zár pontosan, ne nyomjuk, hanem ellenőrizzük, hogy jól tettük-e be a kazettát.

4. Nyomja meg a visszatekerő REWIND gombot a kazettás egységen. Figyelje a kazettát a kis ablakon keresztül, és ha forog, várjon, amíg leáll. Ellenőrizze, hogy a szalag az elején van-e.

5. Nyomja meg a STOP gombot a magnón.

6. Most írja le a következő parancsot:

LOAD "TESZTKARTYA" [RETURN]

Ez összesen 18 leütést jelent, mivel az " is egy leütésnek számít. Ahhoz, hogy a "-et előállítsa, meg kell keresnie

a két [SHIFT] gomb közül az egyiket (bármelyik jó) és lenyomva tartania, amíg leüti a billentyűt.

Ne felejtse elengedni a [SHIFT]-et, amint (de nem korábban) az " megjelenik a képernyőn.

Az üzenetet helyesen kell megadnunk. Néhány elkerülhető gyakori hiba:

– ha úgy írunk, hogy a [SHIFT LOCK] le van nyomva. Furcsa minták jelennek meg vonalakkal, kártyajelekkel és különböző alakzatokkal, és semmi sem fog történni;

– ha két szimpla idézőjelet '' használunk a dupla helyett. A gép ezt fogja válaszolni:

?SYNTAX ERROR (szintaktikus hiba)

READY

Ekkor megpróbálhatjuk újra beírni a parancsot a következő sorba;

– ha hagyunk egy betűhelyet az " és a T, a TESZT és a KARTYA vagy a D és az " között;

– ha leírjuk a RETURN betűket, ahelyett, hogy a [RETURN] feliratú gombot ütnénk le.

Így semmi sem történik:

– ha a Ø-t (nulla) használjuk O (O betű) helyett a LOAD szóban.

Ha hibázunk, a hibát bármikor „kiradírozhatjuk" a [INST DEL] gomb nyomogatásával. Minden leütés kitöröl egy karaktert, és a kurzort egy hellyel visszalépteti.

7. Ha helyesen adjuk meg az üzenetet (még ha hibázunk is a TESZTKARTYA betűinek leírásában), a gép ezt fogja válaszolni:

PRESS PLAY ON TAPE

(nyomja meg a lejátszás gombot a magnón) és ilyen képet mutat:

```
COMMODORE BASIC V3.56Ø671
BYTES FREE
READY
LOAD "TESZTKARTYA"
PRESS PLAY ON TAPE
```

Nyomja le a PLAY gombot a kazettás egységen. A képernyő elsötétedik, és látjuk, hogy a szalag forog. Körülbelül fél perc múlva megjelenik az üzenet:

FOUND TESZTKARTYA
(megvan a TESZTKARTYA)

és a kazetta megáll. Ez azt jelenti, hogy a szalag elért annak a résznek az elejére, ahol a TESZTKARTYA pgrogram van tárolva. Ahhoz, hogy gyorsan betöltsük a programot,

nyomjuk meg a  gombot. (Valójában a program betöltése akkor is elindul, ha bizonyos ideig nem csinálunk semmit.) Amikor megnyomjuk a gombot, a képernyő újra üres lesz, és kb. 3 perc alatt a program bekerül a kazettáról a számítógépbe.

Ha ennyi idő után az az üzenet jelenik meg, hogy:

FOUND AKASZTOTT EMBER

ez azt jelenti, hogy ön hibásan írta le

a TESZTKARTYA szót, amikor eredetileg beírta az üzenetet. Állítsa meg a számítógépet a RUN STOP gomb lenyomásával, és menjen vissza a 4. lépéshez.

Legtöbbször a betöltési folyamat tökéletesen lezajlik. Ha mégsem, és a szalag csak forog és forog anélkül, hogy valami történne, akkor valószínűleg sérült a kazetta. Fordítsuk meg a kazettát, és próbáljuk meg a másik oldalról betölteni a programot. Ha így sem sikerül, szakemberrel ellenőriztessük, ill. javíttassuk meg a kazettás egységet és a számítógépet. Erre aligha lesz szükség, mert minden Commodore számítógép olyan speciális módszert alkalmaz a programok rögzítésére, amely sokkal megbízhatóbb, mint a legtöbb más gyártmány. Ha a TESZTKARTYA programot betöltöttük, a gép azt fogja kiírni:

READY.

Kezdjük a programot azzal, hogy beírjuk:

RUN RETURN

(ez négy billentyű leütése)

Még azt is megtehetjük, hogy lenyomva tartjuk a SHIFT gombot, míg leütjük a f3 f6 billentyűt, és a program magától elindul.

Vége az 1.1 Gyakorlatnak

1.2 Gyakorlat

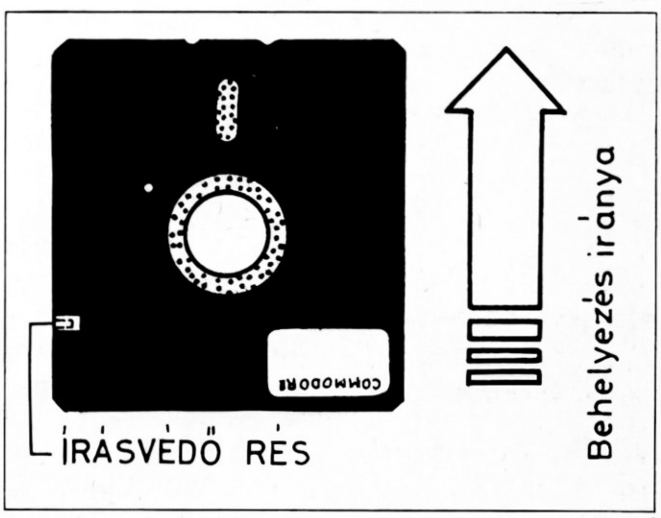
(Lemezegységgel rendelkező olvasóinknak)

Fontos: Olvassa végig az útmutatásokat, mielőtt bármelyiket is kipróbálná.

1. Győződjön meg róla, hogy a lemezegység csatlakoztatva van a hálózathoz és a számítógéphez. Az 1541-es lemezegység hátulján két egyforma foglalat van a csatlakoztatásra, bármelyiket használhatja. Még ne próbálja betenni a lemezt a gépbe!

2. Kapcsolja be a lemezegységet, a hátoldalán levő kapcsolóval. Elöl a zöld és a piros lámpa kigyullad. Egy-két másodpercig a lemezegység ellenőrzi magát, hogy minden helyesen működik-e. Ha minden rendben van, a piros fény kialszik, csak a zöld marad égve. Ha a piros fény nem alszik ki, vagy villogni kezd, ez annak a jele, hogy valamilyen hiba vagy sérülés van. Próbáljuk meg újra ki- és bekapcsolni, és ha a hibajelzés nem szűnik meg, szakemberhez kell fordulnia.

3. Nyissuk ki a lemezegységet úgy, hogy a fogantyút befelé és felfelé nyomjuk. [Újabb egységeknél a kar óramutató járásával ellenkező irányban való vízszintes forgatásával *(a lektor megjegyzése).*] A programot tartalmazó lemezt nyomjuk be határozottan, de finoman a nyílásba, és győződjünk meg róla, hogy teljesen bent van-e. A COMMODORE feliratnak felül kell lenni, az Ön felőli jobb oldalon.



Nyomja meg a fogantyút, amíg előre nem ugrik. [Újabb egységeknél a kart forgassuk vissza függőlegesre *(a lektor megjegyzése).*] E műveletsorral behelyezte a lemezt.

4. Keresse meg a f2 f5 feliratú hosszú billentyűt és nyomja le. Aztán írja le:

TESZTKARTYA

Ekkor a képernyőn meg kell jelennie:

DLOAD"TESZTKARTYA

Ha nem így jelenik meg, bármelyik hibásan beírt karaktert kitörölheti, ha lenyomja a INST DEL gombot. Minden leütés kitöröl egy karaktert, és egy hellyel visszalépteti a kurzort.

Ha biztos benne, hogy az üzenet helyes, nyomja le a RETURN gombot.

Ha mindent helyesen ír, a lemezmeghajtó enyhén kattog egy pár pillanatig, és eközben világít a piros fény. Végül a fény kialszik, és a képernyőn ez az üzenet marad:

COMMODORE BASIC V3.5 6Ø671
BYTES FREE
READY
DLOAD"TESZTKARTYA
SEARCHING FOR TESZTKARTYA
LOADING
READY

Ha ez a folyamat nem így zajlik le, annak két legvalószínűbb oka a következő:

a) Ha nem megfelelően csatlakoztatta a lemezegységet, vagy elfelejtette bekapcsolni, ezt az üzenetet kapja:

DEVICE NOT PRESENT
(az egység nincs jelen)

b) Ha elfelejtette a gépbe tenni a lemezt, vagy rossz irányban tette be, vagy nem megfelelő lemezt helyezett be, ez lesz az üzenet:

FILE NOT FOUND
(nincs meg az állomány)

Mindkét esetben keresse meg a hibát, korrigálja és próbálja meg újra.

Ha a programot helyesen betöltötte, elkezdheti leírni:

RUN RETURN

(4 billentyű leütés). Az is jó megoldás, ha lenyomva tartja a SHIFT -et, és leüti az f3 f6 billentyűt: a program beindul.

Mielőtt továbbmennénk, felhívjuk a figyelmét a lemez kezelésével kapcsolatos további néhány fontos dologra:

– Soha ne kapcsolja ki vagy be a lemezegységet, amíg benne van egy lemez. Mindig a bekapcsolás után tegye be, és a kikapcsolás után vegye ki a lemezt.

– Minél kevesebbet legyen kézben a lemez; óvjuk a portól, a hőségtől, a hidegtől, a nedvességtől, és soha ne fogjuk meg a felületét ott, ahol a papírborítóból kilátszik.

– Tartsuk távol a lemezeket minden elektromos berendezéstől, mint pl. tv, villamos motorok, fém detektorok. Ha a lemezt egy repülőtéri biztonsági ellenőrzésen kell keresztülvinnie, ne hagyja, hogy megröntgenezzék.

(A kazettás egységgel rendelkező olvasók itt ismét csatlakozhatnak.)

Az első programból megismerhetjük a számítógéppel kialakítható színek és hangok skálájának egy részét. A program futása közben a tv-készülék beállításával elérhetjük a legjobb színminőséget. Ne felejtse el felhangosítani a tv-készüléket, hogy hallja is a program által keltett hangokat! Ha már eleget nézte a TESZTKARTYA-t, leállíthatja a programot a RUN STOP gomb lenyomásával. Amikor lenyomja ezt a gombot (vagy amikor a program bármilyen okból leáll), ilyen üzenet jelenik meg a képernyőn:

BREAK IN 56Ø
(megszakítás az 56Ø-ban)

READY

A példánkban álló 56Ø bármilyen szám lehet. A BREAK nem azt jelenti, hogy a számítógép eltörött [break eredeti jelentése törés, eltörni (*a lektor megjegyzése.*)] csak azt jelzi, hogy a programot alkotó utasítások sorát megszakítottuk, az utasítást megtörtük.

Vége az 1.2 Gyakorlatnak

1.3 Gyakorlat

Az 1.3 Gyakorlat szókitaláló játék, amit azért terveztünk, hogy segítsük a használót a billentyűzet megismerésében. A program címe AKASZTOTT EMBER. Betölthető kazettáról is, lemezről is.

A kazettáról:

LOAD "AKASZTOTT EMBER" RETURN

ill. a lemezről:

DLOAD "AKASZTOTT EMBER RETURN

paranccsal.

Ne felejtse el, hogy az f2 f5 gomb lenyomása automatikusan adja a DLOAD" sorozatokat.

Ha a programot betöltötte, és megjelenik a READY üzenet, írja le:

RUN RETURN vagy

nyomja meg az f3 f6 gombot, közben lenyomta tartva a SHIFT -et, és a játék magától elkezdődik. Ha nem ismeri a játékot, próbálja egyszerűen a betűket nyomogatni, és figyelje (és hallgassa), hogy mi történik. Hamarosan rá fog jönni a játék lényegére.

Játssza ezt a játékot, amíg kedve tartja, és használja ki a lehetőséget arra, hogy megismerkedjék a billentyűzet betűinek használatával.

Vége az 1.3 Gyakorlatnak

2. LECKE

Ez a lecke a billentyűzettel ismertet meg. Bemutatja, hogyan küldhetünk a gépnek üzeneteket a billentyűzetről, ill. hogyan rajzolhatunk ábrákat a képernyőre. Nem lesz egészen ismeretlen a billentyűzet, ha használt már egyszerű írógépet. A betűket, a számokat és a jelek nagy részét az írógépen megszokott helyen találja, ott van a SHIFT (váltó) és a SHIFT LOCK (váltózár) billentyű is, bár a számítógépen ez egy kicsit másképp működik.

Akkor se keseredjen el, ha még soha nem írt írógépen. Ebben az esetben egy kicsivel több időt kell fordítania a billentyűzet megismerésére.

Ebben a leckében ne használja a RETURN gombot, hacsak külön nem mondjuk. Mint azt az 1. leckében láttuk, ezzel a billentyűvel lehet elérni, hogy a gép ténylegesen végrehajtson egy parancsot, mint például a program betöltését. Pillanatnyilag csak a képernyőt használjuk, és nincs szükség a számítógép segítségére. Ha mégis megnyomja a RETURN -t, a számítógép megpróbál engedelmeskedni a beírt üzenetnek vagy képnek, és ha nem tudja azt értelmezni, elrontja a képernyőre rajzolt ábrát. A másik jel, amit most ne használjunk, a dupla idézőjel ("). Ennek a jelnek sajátos jelentése van, ha leütjük, néhány billentyű hatása egészen más lesz, mint idézőjel nélkül. Ha pl. megjelenik a képernyőn a dupla idézőjel, sokkal nehezebb használható rajzokat készíteni. Egy későbbi leckében majd megtudunk mindent erről a karakterről, de most hagyjuk békén.

Lehet, hogy megrémül ettől a sok „ne" kezdetű tiltó mondattól. Még egy „ne": ne aggódjon! Ez a számítógép nem olyan, mint

a sci-fi regények gépei, nincs benne „pusztítsd el magad" parancs. Egyszerűen nem lehet elrontani a gépet azzal, hogy írunk a billentyűzeten. A gép elég furcsán viselkedik az olyan szövegek begépelésekor, amelyekben benne van az " vagy a RETURN, és ha véletlenül vagy figyelmetlenségből leüti pl. az utóbbit, a számítógép abbahagyja a beírt karakterekre való reagálást. Ezek csak időszakos gondok: helyre lehet őket hozni a számítógép ki-, majd újra bekapcsolásával vagy a főkapcsoló melletti Reset (alaphelyzet) gomb lenyomásával.

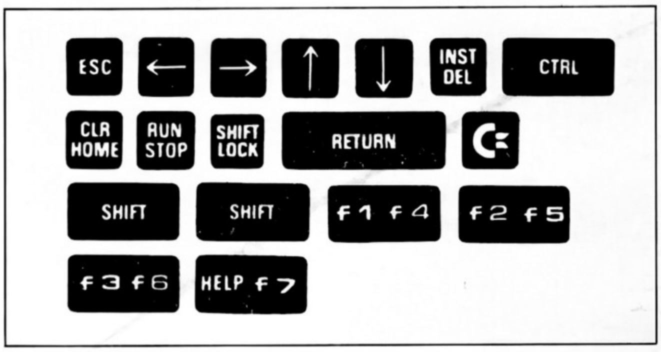
2.1 Gyakorlat

A Commodore 16 billentyűzetén 66 billentyű van, a Plus/4-esén 67. Bemutatjuk mindkét billentyűzet képét.
A billentyűk kétfélék:

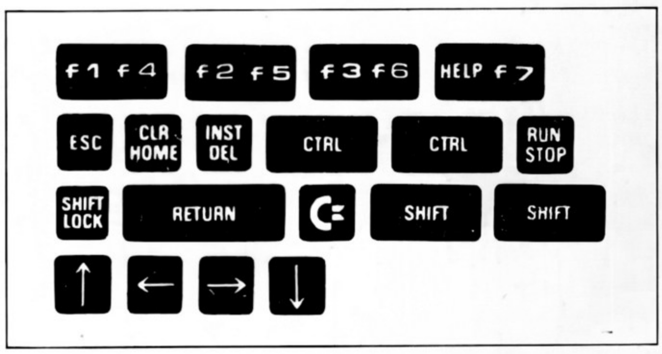
– 48 'jel' billentyű, amelyekkel a gép karaktereket ír ki a képernyőre. (A 'space' – szóköz – jel is billentyűnek számít.)

– 18 (vagy 19) 'funkció' billentyű, amelyek a karakterek megrajzolásának módját vezérlik.

A COMMODORE 16 funkcióbillentyűi:



A COMMODORE Plus/4 funkcióbillentyűi:



A billentyűk elrendezésén kívül az az egyetlen különbség, hogy a Plus/4-en a Control (CTRL, vezérlő) billentyű kétszer szerepel.

Nyomjuk meg többször a SHIFT LOCK billentyűt, és figyeljük meg, hogy két állása van: fent és lent. Végül győződjünk meg róla, hogy a 'fent' helyzetben hagytuk. Indítsuk be a gépet a szokásos módon, vagy nyomjuk le a Reset gombot, ha már be volt kapcsolva. Pontosan a READY üzenet alatt látjuk a villogó kurzort.

Ebben a gyakorlatban azt vizsgáljuk, hogyan mozog a kurzor, miközben a jelek rajzolódnak a képernyőn. A kurzor megmutatja, hogy hol fog megjelenni

COMMODORE 16



COMMODORE Plus/4



a következő leírt karakter. Írjon le néhány karaktert, és figyelje a kurzort. Figyelje meg, hogy minden karakter a kurzor *helyére* kerül, míg az továbblép a következő pozícióba.

Írjon tele betűkkel egy egész sort a képernyőn, amíg a kurzor el nem kerül a fehér felület legvégéig. Írjon le még egy betűt, és figyelje meg, mi történik: a kurzor magától a következő sor elejére ugrik.

Mielőtt továbbmenne, számolja meg a betűket a képernyő teljes szélességében, és írja be a keretbe:

[ ] karakter hely van a képernyő minden sorában.

Aztán írjon újabb sorokat, amíg el nem éri a képernyő alját. Ez nem nagy feladat, mert ha bármelyik jelet lenyomva tartja, 'ismételni' fogja magát, és hamarosan megtölti a képernyőt. Számolja meg a keletkezett sorokat, és írja be a lenti keretbe. Ne felejtse el beszámítani az üres sorokat is, amelyek az eredeti

COMMODORE BASIC ...
üzenet fölött és alatt vannak.

[ ] sor van a képernyőt betöltő karakterekből.

Most töltse ki az utolsó sort, amíg a kurzor el nem éri a képernyő jobb alsó sarkát. Üssön le még egy billentyűt, és figyelje a kurzort. Az egész képmező felfelé mozdul, a kurzor pedig a képernyő alján levő következő üres sor elejére ugrik. Minden újabb alsó sor megjelenésével eltűnik a legfelső sor. A felső sor már nincs, és nem is lehet visszahozni, hacsak nem tárolta valahol máshol.

Töltsön ki még néhány sort, és győződjön meg arról, hogy a rendszer az újabb szöveg számára mindig biztosít helyet a képernyő alján.

Vége a 2.1. Gyakorlatnak

2.2. Gyakorlat

(Az alábbiak a Commodore 16-ra vonatkoznak.)

A számítógépnek 48 db jel billentyűje van, de sokkal több különböző jelet tud kiírni, köztük betűket, számokat, különleges karaktereket, matematikai jeleket és egyszerű vagy 'grafikai' alakzatok széles skáláját, amelyek kombinációjából különböző képeket lehet összeállítani. Mindezekből a karakterekből a két SHIFT billentyű bármelyikének használatával lehet választani (a számítógép belsejében össze vannak kapcsolva), valamint a speciális C= feliratú 'Commodore' billentyűvel. Indítsuk újra a gépet, és írjuk le:

1234567890QWERTYUIOP@+ –
ASDFGHJKL:;*ZXCBV

(Ezek a soronként vett jel billentyűk.)

Most nyomjuk le az egyik SHIFT billentyűt, és írjuk le újra a sort. Teljesen más jelek sorát kapjuk (köztük az "-et, de ez nem fog gondot okozni, ha betartja az utasításainkat). Másolja le az alábbi táblázat második sorába a jeleket, és nézze meg, mennyire összeillik néhánynak (például az U-n és I-n lévőnek) a grafikája. Figyelje meg, hogy a grafikus jelek általában az azokat befoglaló kis négyzeteknek a széléig érnek, ezért egymást érinthetik. Gépeljük le a sort harmadszor is, de közben tartsuk lenyomva a C= billentyűt. Most megint sok jel megváltozik. Másoljuk be ezt a sort a táblázat harmadik sorába.

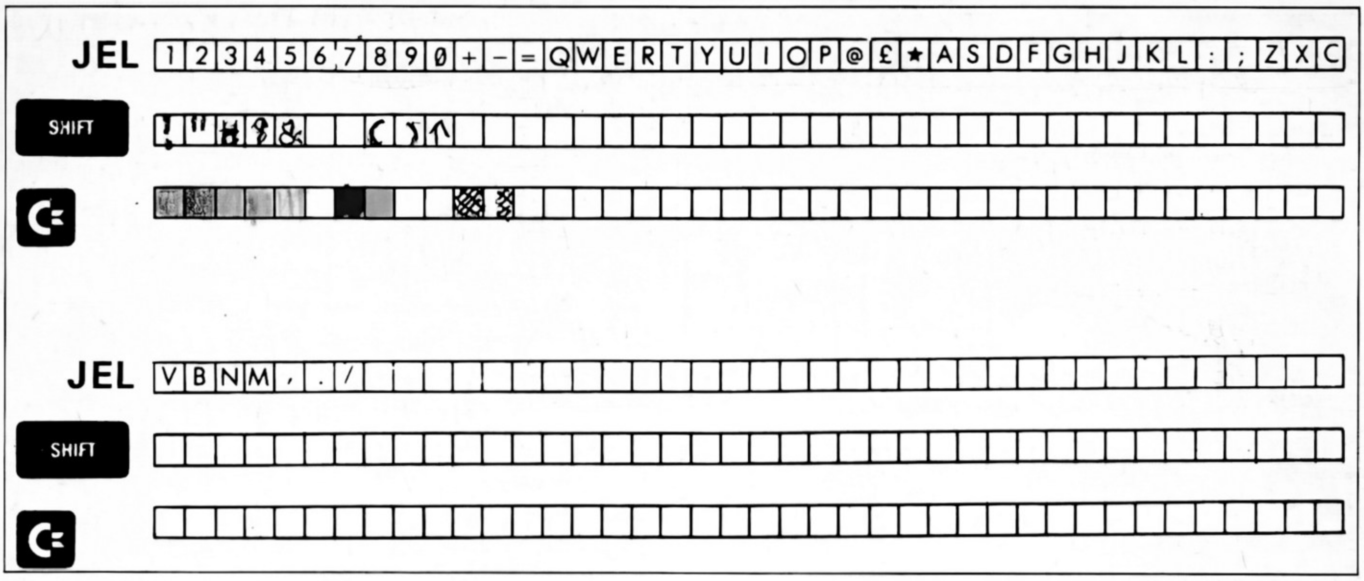
A grafikus jelek további vizsgálatához ismételje meg a gyakorlatot az NM,./£=sorral, majd folytassa 33-szor a szóközbillentyűvel, és így a képernyő széléig fog jutni.

Jegyezze meg, és emlékezzen arra, hogy a Ø (nulla) jel más, mint az O (O betű). Önnek mindig Ø-t (nullát, vonallal áthúzottat) kell használni, ha a számra van szüksége, és nem a betűre.

Nyomja le egyszerre a C= és a SHIFT gombokat. A képernyőn a nagybetűk átalakulnak kisbetűkké, míg néhány grafikai jel átalakul nagybetűvé. [A Tudományszervezési és Informatikai Intézet által forgalmazott gépek esetében ékezetes kis- és nagybetűk is megjelennek. Lásd a Felhasználói Kézikönyvet *(a lektor megjegyzése).*] Nyomja meg újra a két gombot egyszerre (vagy tartsa lenyomva), és visszatérnek a nagybetűk. Általában használhatja a grafikai jelek teljes készletét vagy a kisbetűket és a grafikus jelek egy részét, de a kettőt egyszerre nem! A kisbetűk használatával ennek a sorozatnak a második kötetében foglalkozunk. [A kísérő programok magyar változata alkalmazza a kis- és nagybetűket, mivel az

JEL	1	2	3	4	5	6	7	8	9	Ø	Q	W	E	R	T	Y	U	I	O	P	@	+	–	A	S	D	F	G	H	J	K	L	:	;	★	Z	X	C	V	B
SHIFT																																								
C=																																								

JEL	N	M	,	.	/	£	=																																	
SHIFT																																								
C=																																								



ékezetes betűk csak akkor jelennek meg. Ezért a programok futtatásakor erre szükség lesz annak ellenére, hogy a tanfolyam első részében nem szerepel *(a lektor megjegyzése).*]

(Az alábbiak a COMMODORE Plus/4-re vonatkoznak.)

A számítógépnek 48 jel billentyűje van, de sokkal több különböző jelet tud alkotni; köztük betűket, számokat, különleges karaktereket, matematikai jeleket és olyan egyszerű vagy 'grafikai' alakzatok széles skáláját, amelyek kombinációjából különböző képeket lehet összeállítani. Mindezekből a karakterekből

a két SHIFT billentyű bármelyikének (ezek a számítógép belsejében össze vannak kapcsolva), valamint

a speciális C= feliratú Commodore billentyű használatával lehet választani.

Indítsa újra a gépet (vagy nyomja meg a Reset gombot), és írja le:

1234567890+–=QWERTYUIOP@£* ASDFGHJKL:;ZXC
(Ezek a soronkénti jel billentyűk.)

Nyomja le az

egyik SHIFT billentyűt, és írja le a sort újra. Majdnem teljesen más jelek sorát kapja, köztük az "-et (de ez nem fog gondot okozni, ha betartja utasításainkat). Másolja le az alábbi táblázat második sorába a jeleket, és figyelje meg, hogy néhány alak mennyire összeillik (például az U-n és az I-n lévők). Észreveheti, hogy a grafikai jelek általában elérik annak a kis négyzetnek a széleit, amelyet elfoglalnak, ezért össze tudnak érni.

Gépelje le harmadszor is a sort, de

közben tartsa lenyomva a C= billentyűt. Megint sok jel megváltozik. Másolja be az így kapott sort a lenti táblázat harmadik sorába.

Ahhoz, hogy megvizsgálja a többi grafikai jelet, ismételje meg a gyakorlatot a VBNM,./ sorral, majd írjon le 31 szóközt (space), ezzel eljut a képernyő széléig. Ne felejtse el, hogy a Ø szám különbözik az O betűtől. Mindig az áthúzott Ø-t használja, ha a számra gondol, és nem a betűre.

Nyomja le egyszerre a C= és

a SHIFT billentyűket. A nagybetűk a képernyőn kisbetűkké válnak, miközben néhány grafikai jel átalakul nagybetűvé. Nyomja meg újra együtt a két billentyűt (vagy tartsa őket lenyomva), és a nagybetűk visszatérnek. Általában használhatja a teljes grafikai jelkészletet vagy a kisbetűkészlet egy részét, de a kettőt egyszerre nem! A kisbetűk használatát ennek a sorozatnak a második kötetében tárgyaljuk.

Vége a 2.2 Gyakorlatnak

2.3 Gyakorlat

Eddig arra szorítkoztunk, hogy szigorú sorrendben, balról jobbra, fentről lefelé haladva helyezzük el a karaktereket. Így nagyon unalmas egy képet megrajzolni. Mennyivel kényelmesebb, ha ott helyezhetjük el a szöveget és a grafikai jeleket, ahol akarjuk. Ezt az öt kurzor-léptető billentyűvel lehet elérni:

[CLR HOME] [←] [→] [↑] [↓] (A Plus/4-en a nyíllal jelölt gombok ténylegesen nyíl alakúak.)

Ha csak a [CLR HOME] billentyűt nyomjuk le, a kurzor 'haza'-megy [HOME = otthon, haza *(a lektor megjegyzése)*] a képernyő bal felső sarkába. Hozza alapállapotba (RESET) a számítógépet, és nyomja le ezt a billentyűt. Látni fogja, hogy a kurzor visszamegy a képernyő sarkába.

A négy nyíllal jelölt billentyű segítségével a nyíl irányába mozgathatja a kurzort. Ha most lenyomja a [↓] -t, a kurzor egy sorral lejjebb ugrik a COMMODORE C betűjére. A betű továbbra is látszik, mert a kurzor átlátszó. Próbálja a vonalban tovább mozgatni a kurzort a [→] gombbal, aztán mozgassa vissza a kiinduláshoz a [←] billentyűvel. A sorban egyetlen karakter sem fog megváltozni, amíg csak a kurzor billentyűket használja.

Mozgassa a kurzort a képernyőn egy karakterre. Ha most leüt egy jel billentyűt, az az új karakter jelenik meg, és a kurzor előrelép egy hellyel. Ha például a kurzort a COMMODORE C-jére állítja, és leírja, hogy NAGYSZERU, a sor így fog kinézni:

NAGYSZERU BASIC ...

(Természetesen bármilyen 9 betűs szóval elvégezhető a gyakorlat.)

A [→] billentyűvel mozgassa el a kurzort a BYTES szó B betűjére, és alakítsa át a szót EMBER-re.

Mint az összes többi billentyű, a kurzor vezérlő billentyűk is ismétlik önmagukat, tehát ha az egyiket lenyomva tartjuk, a kurzor körülbelül 10 karakterhelyet lép folyamatosan egy másodperc alatt. Jó ezt tudni, ha a kurzort gyorsan akarjuk mozgatni.

Ha akkor nyomjuk meg a [→] billentyűt, amikor a kurzor a sor végén van, a kurzor átugrik a következő sor elejére.

Hasonlóképpen a sor elején a [←] billentyű a kurzort az előző sor végére mozgatja.

A [↑] és [↓] gombok egyenesen felfelé, ill. lefelé mozgatják a kurzort, oldalmozgás nélkül.

A képernyő legalját és legtetejét sajátosan kell kezelni, ezért érdemes megfigyelni, mi történik.

Írjuk tele a képernyőt néhány sorral, és mozgassuk el a kurzort az utolsó sor végéig. Mikor túlmozgatjuk a kurzort, ennek a sornak a végén a [→] vagy a [↓] billentyűvel, az egész képernyő feljebb lép, mintha egy új karaktert írtunk volna be. A legfelső sor pedig eltűnik.

Most menjünk 'haza' a bal felső sarokba, és próbáljuk visszafelé mozgatni a kurzort. A képernyő nem lép lefelé, mint ahogy gondoltuk volna; semmi sem történik, és a kurzor a helyén marad.

Most nyomjuk meg a [CLR HOME] -t, miközben lenyomva tartjuk az egyik [SHIFT] -et. A kurzor 'hazamegy', a képernyő letörlődik, így tiszta képernyőn dolgozhatunk tovább.

Gyakorolja a rajzkészítést a képernyőn a grafikai jelek és a kurzor vezérlő billentyűk segítségével. Kezdje olyan egyszerű alakzatokkal, mint a négyzet, a háromszög vagy a kis körök.

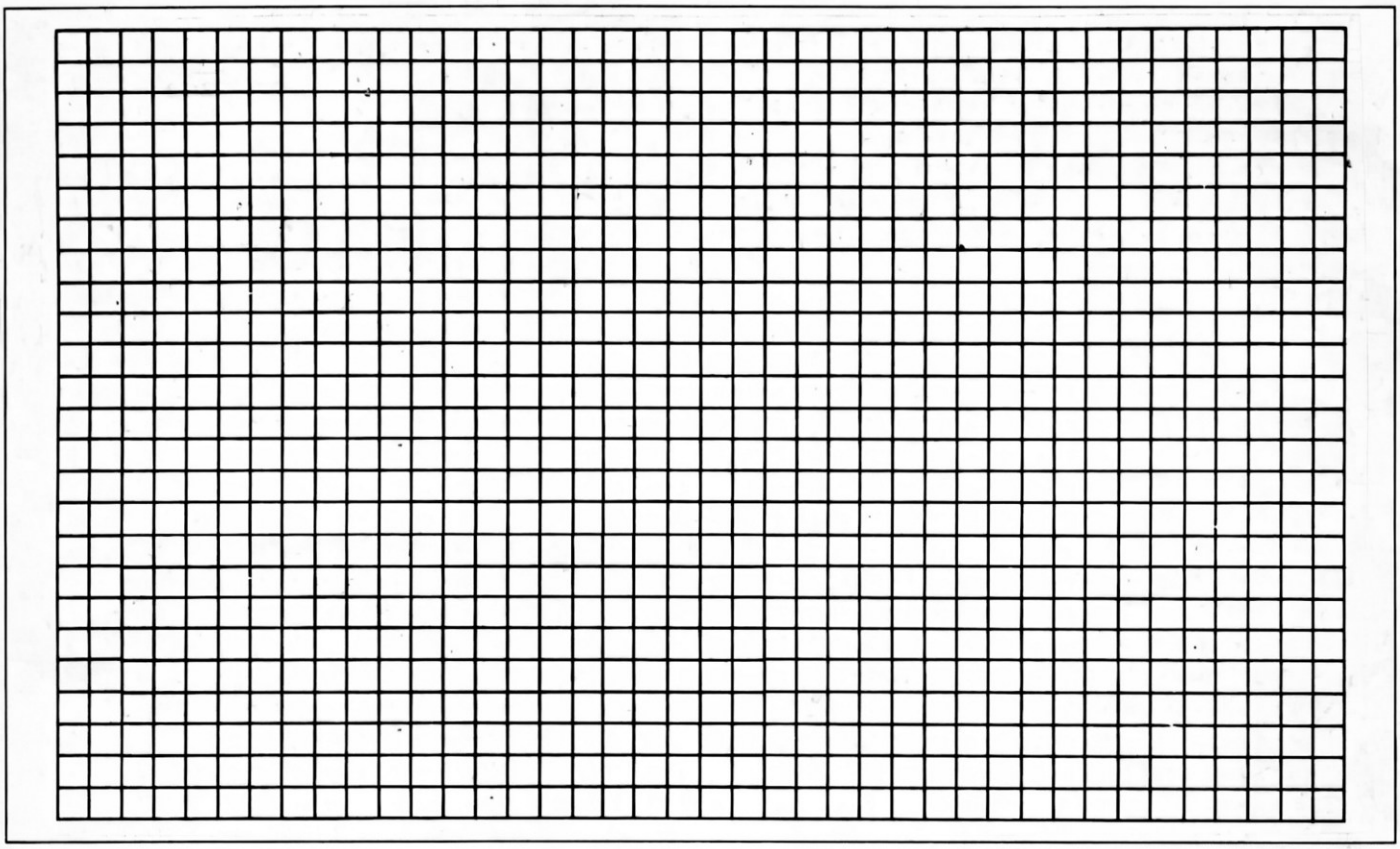
Ha hibát vét, léptesse vissza a kurzort, és írja le helyesen a karaktert.

A [SPACE] billentyű lenyomásával lehet eltüntetni a rossz helyen álló karaktereket. Amikor úgy érzi, hogy hozzászokott a billentyűzethez, rajzoljon egy ilyen keretet, és írja bele a nevét.

VARJAS
ANTAL

Vagy pl. rajzoljon fel kártyalapokat legömbölyített sarokkal, és a megfelelő jelekkel (azt javasoljuk, hogy 10-et vagy annál kevesebbet érő lapokkal próbálkozzon). Végül, ha művészi tehetsége is van hozzá, próbáljon állatokat, űrhajót vagy emberi arcot rajzolni.

Az alábbi rácshálót használva tervezze meg előbb a rajzot:



Vége a 2.3 Gyakorlatnak

2.4 Gyakorlat

Mindenki hibázik gépelés közben. Ha egyetlen betűhibát ejt egy szó közepén, kijavíthatja a kurzor vezérlő gombokkal. Ha pl. AUSXTRIA-t ír AUSZTRIA helyett, vissza lehet léptetni a kurzort az X fölé, és átírni Z-re. Próbálja meg!

Ha nem megfelelő számú betűt írt le (hanem a szükségesnél többet vagy kevesebbet), ez a módszer sajnos nem használható. Sokkal hatékonyabb ennél a [INST DEL] (beszúrni, törölni) billentyű, amellyel be lehet szúrni vagy törölni lehet a karaktereket a képernyőről.

Ha önmagában nyomja le a [INST DEL] -t, kitörli a kurzor bal oldalán lévő karaktereket, és a sorban levő összes többi karaktert eggyel balra lépteti, így kitölti az üres helyet.

Tegyük fel, hogy hibásan azt írta le, hogy INXDIA, pedig INDIA-t akart. El akarja hagyni az X-et, tehát mozgassa a kurzort a D-re (nem az X-re), és nyomja le a [INST DEL] -t. Az X eltűnik, és a DIA eggyel hátrább mozdul, így INDIA marad, s a szóban nincs üres hely.

Most próbálja használni a [INST DEL] billentyűt javításokra,

pl.: KAINA-t KINA-ra
EEGYIPTOM-ot EGYIPTOM-ra
BATUMI-t BATU-ra
AUSZTRALIA-t AUSZTRIA-ra

A gyakorlatban leginkább arra használjuk ezt a billentyűt, hogy olyan karaktert töröljünk vele, amit éppen akkor írtunk le. A billentyű egy lépésben törli az utolsó jelet, és visszaállítja a kurzort. Hamarosan megszokja majd, hogy a [INST DEL] billentyűt kell használnia, ha hibázik a gépelésben.

Az [INST DEL] másik alkalmazása, ha felső állásban használjuk, azaz lenyomjuk a [SHIFT] -et az [INST DEL] leütése közben. Ebben az üzemmódban arra használjuk, hogy helyközöket szúrjunk be a szavak vagy sorok közepébe. Ezeket a helyközöket aztán a szokásos módon lehet karakterekkel megtölteni.

Próbálja meg a következő feladatot, amelyben az AUSZTRIA szót változtatjuk AUSZTRALIA-vá.

Töröljük le a képernyőt ([SHIFT] és [CLR HOME]), és írjuk le:

AUSZTRIA

Lépjen a kurzorral vissza az I-re.

Nyomja le a [SHIFT] billentyűt, és üsse le a [INST DEL] -t kétszer. Az IA minden leütéskor egy hellyel jobbra lép, így két ugrás után ezt kapjuk:

AUSZTR ■ IA

Most szúrja be az AL-t. Mozgassa el a kurzort a szó végén túlra a [→] billentyűvel (ne a helyköz gombjával, mivel az kiradírozna olyan karaktereket is, amelyekre még szükségünk van).

Az [INST DEL] használatának gyakorlására töröljük le a képernyőt, írjuk be a bal oldali oszlop szavait, majd alakítsuk át őket a jobb oldali oszlopban álló szóvá.

HOTEL	MOTEL
MIKROFON	MIKROCOMPUTER
ESZIK	ISZIK
KULCS	KULACS
RAGÁLY	DAGÁLY
KICSI	KOCSI
LOVAGOL	LOVAGIAS
PACSI	CSACSKA
VALAMI	VALLAT
HAGYMA	HAGYJ MA
JAVA	JAMAICA
ELEJE	VELEJE
TAXI	TAXAMETER
CSIKOS	PISZKOS
KOLOMP	KOLONC
PORTA	RAPPORTRA
EREDET	EREGET
HOLD	HOLLAND
KEREST	KEVESET
ERRE	EMERRE
ADAT	FELADAT
PAPOK	PADOK

És most alakítsuk őket vissza!

Vége a 2.4 Gyakorlatnak

A 2. leckéhez tartozó program címe GYORSGEPELES. Segít a billentyűzet megismerésében. Töltse be a gépbe úgy, hogy leírja:

```
LOAD "GYORSGEPELES"
```

vagy

```
DLOAD "GYORSGEPELES
```

Indítsa el a programot a RUN paranccsal, és addig gyakoroljon, amíg szükségesnek érzi.

3. LECKE

Mind a Commodore 16, mind a Plus/4 színes megjelenítésre alkalmas számítógép. Ez a lecke tartalmazza azokat az ismereteket, amelyek birtokában színes képeket tudunk a képernyőre rajzolni.

Ha fekete-fehér készüléke van, ne várjon túlzottan nagy sikereket a 3. leckétől! Ennek ellenére végezze el a feladatokat.

3.1 Gyakorlat

Az 1. lecke TESZTKARTYA programjával ellenőrizze, hogy a tv-készüléke jól van-e beállítva.

A RUN STOP billentyű megnyomásával állítsa le a programot. Azt fogja tapasztalni, hogy a kurzor fekete. Azonban a kurzor színét is tudja változtatni, s ahogy megmutatja, hol lesz a következő karakter, azt is jelöli, hogy az *milyen színű* lesz. Próbáljon leírni egy sorozatot a ▄ jelből (C= és I billentyűk), és figyelje meg, hogy fekete színben íródnak ki, mert pillanatnyilag a kurzor fekete.

A kurzor színét tetszés szerint megváltoztathatjuk, ha leütjük a nyolc színbillentyű valamelyikét, miközben lenyomva tartjuk a CTRL gombot. A színbillentyűk 1-től 8-ig vannak számozva, és rajtuk van az általuk vezérelt szín rövidítése is. Tartsa lenyomva a CTRL billentyűt, és nyomja le egymás után a színvezérlő gombokat.

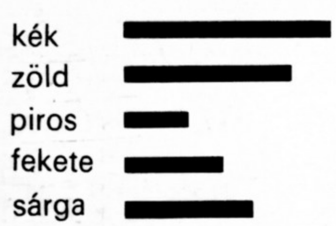
Ha lenyomja az 1-es (BLK) billentyűt, a kurzor színe *feketére* változik (hacsak nem

volt előtte is fekete). A 2-es (WHT) gomb lenyomásakor a kurzor *fehérre* változik. A többi gomb a számok sorrendjében *pirosra, türkizkékre, bíborra, zöldre, kékre* és *sárgára* változtatja a kurzor színét. Amikor a kurzor színe fehér, nem látszik a fehér háttéren, olyan, mintha eltűnt volna. Egy nyolc színből álló másik színskálát kapunk, ha a CTRL helyett a C= billentyűt használjuk. Ezek kevésbé élénk, pasztell árnyalatok:

- C= és 1 narancs
- C= és 2 barna
- C= és 3 sárgászöld
- C= és 4 rózsaszín
- C= és 5 sötétzöld
- C= és 6 világoskék
- C= és 7 sötétkék
- C= és 8 világoszöld

Próbáljon meg színes képeket készíteni. Kezdésnek nagyon jó, ha különböző hosszúságú színes oszlopokat rajzol. Az oszlop megszerkesztéséhez használja a grafikai jelet (C= és U). Egy 5 jel hosszú piros oszlop megszerkesztéséhez például nyomja le először a CTRL -t, majd a 3-as (RED) billentyűt, ettől a kurzor pirosra változik. Ekkor nyomja le a C= -t, és üsse le ötször az U-t.

A kurzor színét bármikor meg lehet változtatni. Amikor már megismerkedett a színvezérlő billentyűk használatával, próbáljon megrajzolni egy olyan „választási eredmények” ábrát, mint amilyet az alábbi ábrán lát. A betűket hagyja meg feketén, hogy ezzel is kiemelje az oszlopok színeit.



Vége a 3.1 Gyakorlatnak

3.2 Gyakorlat

Azt már láthatta, hogyan kezeli a számítógép a képernyőn az egyes karakterek színeit. A képernyő keret és a háttér színárnyalatának vezérlésére is van lehetőség. A gépen nincs olyan billentyű, amely kizárólag ezeknek a színeknek a vezérlését végezné, ezért csak külön parancsokkal változtathatjuk azokat.

Írja le: COLOR 4,5. Ellenőrizze az írás helyességét alaposan, szükség esetén javítsa ki, majd nyomja le a RETURN billentyűt. A tv-képernyő kerete azonnal bíbor színűvé változik.

Ez a speciális parancs három részből áll:
COLOR: ez a kulcsszó.

4: ez a szám mondja meg a számítógépnek, hogy a képernyő melyik részén kell a változásnak megtörténnie. Azaz COLOR Ø, ... a háttér színét változtatná meg a képernyőn, COLOR 1 a kurzorra vonatkozna és COLOR 4 a karakterre.

5: ez egy színkód

Az egész táblázat így néz ki:

Szín	FEKETE	FEHÉR	PIROS	TÜRKIZKÉK
Kód	1	2	3	4
Szín	BÍBOR	ZÖLD	KÉK	SÁRGA
Kód	5	6	7	8
Szín	NARANCS	BARNA	SÁRGÁS-ZÖLD	RÓZSA-SZÍN
Kód	9	10	11	12
Szín	SÖTÉT-ZÖLD	VILÁGOS-KÉK	SÖTÉTKÉK	VILÁGOS-ZÖLD
Kód	13	14	15	16

Most már érthető, hogy a "COLOR 4,5" parancs miért változtatta a keretet bíbor színűre. Próbáljon ki néhány hasonló parancsot, mint például:

COLOR 4,2 RETURN

COLOR 4,14 RETURN

A képernyő háttérszínét hasonlóképpen lehet megváltoztatni, a COLOR Ø,... használatával. Próbálja meg például:

COLOR Ø,8

Ha a COLOR parancs leírása közben hibázik, a következő üzeneteket kaphatja:

?SYNTAX ERROR, ha szintaktikus hiba van a parancsban, vagy

?ILLEGAL QANTITY ERROR (nem megengedett mennyiség), ha a számok nem jó intervallumban vannak (Ø–4 az elsőre, 1–16 a másodikra).

Egy kép rajzolásakor célszerű, ha először beállítja a keret- és háttérszínt a megfelelően kiválasztott COLOR parancsokkal, majd a képernyőt letörli (CLR HOME és SHIFT). Ha ugyanarra a színre állítja be a háttért és a kurzort, a kurzor nem látható. Bizonyos színkombinációk esetén is (pl.: sárga és narancs) lehetetlen a karaktereket a háttértől megkülönböztetni. Ha „elveszti" a kurzort, a RESET gomb megnyomásával mindig visszaszerezheti.

Vége a 3.2 Gyakorlatnak

3.3 Gyakorlat

Lehet, hogy már észrevette, hogy furcsa hiányok vannak a grafikai jelek között. Van például , de nincs : van , de nincs és vagy . Úgy látszik, nem lehet egy teljes négyzetet színnel kitölteni, így aztán azonos árnyalatú nagyobb felületeket sem tudunk alkotni.

Itt jön segítségünkre az *inverz mező* lehetősége. Amikor egy karakter inverz mezőben jelenik meg, a karakter és a háttér színei felcserélődnek. Próbálja meg a következő kísérletet:

Hozza alapállapotba a számítógépet, és írjon le néhány karaktert, köztük , , és a szóköz is. Most nyomja le a CTRL -t és a 9-es billentyűt (amire ez is rá van írva: RVS ON). Ezután engedje el a CTRL -t és írjon le még pár karaktert. Ezek fehéren fognak megjelenni a fekete háttéren, nem pedig feketén a fehér háttéren. Természetesen a így jelenik meg ,

a és átalakul és -ra, a szóköz pedig tömör fekete kocka lesz. Ahhoz, hogy a következő karaktereket visszaállítsuk az eredeti alakba, nyomjuk le a CTRL -t, és üssük le a Ø billentyűt (amelyen az RVS OFF felirat is áll).

A kurzor nem mutatja, hogy a gép normál vagy inverz üzemmódban működik. Ha sok inverz jelet használunk, könnyű elfelejteni, hogy hol vagyunk, és gondolkodhatunk rajta, hogy vajon hol jelenik meg a következő karakter. Ezt a gondot kétféleképpen oldhatjuk meg:

a) Írja le a következő karaktert és nézze meg. Ha rossz, törölje ki, változtassa meg az üzemmódot, és kezdje újra.

b) Nyomja le a RVS ON vagy RVS OFF billentyűt attól függően, hogy melyik a helyénvaló, mielőtt leírná a következő jelet. Ha a gép már a megfelelő üzemmódban van, ez a billentyűnyomás semmi változást nem jelent.

Ha a képernyőt színblokkokkal kívánjuk megtölteni, célszerű az inverz mezejű szóközt használni. A szóköz billentyű 'önismétlő', és ha lenyomva tartjuk, másodpercenként körülbelül 1Ø szóközt fog kiírni. A nemzetek zászlóinak megrajzolásával is jól lehet gyakorolni a színek használatát. Azt a zászlót a legkönnyebb reprodukálni, amelyik vízszintes csíkokból áll, mint pl. Luxemburgé.

bíbor
sárga
zöld

Hogyan kezdjünk hozzá? Először állítsuk be megfelelően a keret színét (mondjuk feketére), majd a háttér színét a zászló jobb alsó sarkának színére (zöldre). Ezt így éri el:

COLOR 4,1 RETURN

COLOR Ø,6 RETURN

Aztán vigye 'haza' a kurzort, törölje le a képernyőt, válassza ki a bíbort inverz üzemmódban (nyomja le a CTRL -t, üsse le az 5 PUR -t, aztán a RVS ON-t). Tartsa lenyomva a szóköz gombját, és töltsön meg 8 sort inverz bíbor szóközökkel.

Ezután válassza ki a sárgát, és töltsön meg 9 sort sárga inverz szóközökkel.

Végül váltsa a színt zöldre. Ettől a kurzor eltűnik, és egy 8 soros zöld mezőt hagy a zászló alján. Fontos, hogy a háttérszínt olyanra állítsa, mint a képen megjelenő színek egyike, mert különben nem tudja eltüntetni a kurzort, amikor a rajz készen van. Viszont a keretszínnek különböznie kell minden színtől, amely a zászlót alkotja.

Amikor már elsajátítottuk a vízszintes osztású zászlók megrajzolását, próbálkozzunk függőleges osztásúakkal, mint például az olasz zászló. Nem haszontalan keresztes zászlókat rajzolni (Svájc vagy Izland).

Még nehezebbek a diagonális osztású zászlók, mint pl. Csehszlovákiáé. A dőlt vonalak mentén lévő részeket

a  vagy  grafikai jelekkel kell megrajzolni, ha szükséges, inverzbe fordítva. Ha elgondolkodik rajta, rá fog jönni, hogy háttérszínnek azt kell választani, amelyik a ferde vonal egyik oldalán van.
A csehszlovák zászló esetében a kék a megfelelő háttérszín, a piros például nem volna jó, mert nem tudnánk berajzolni a kék és a fehér részeket a felső diagonálba.

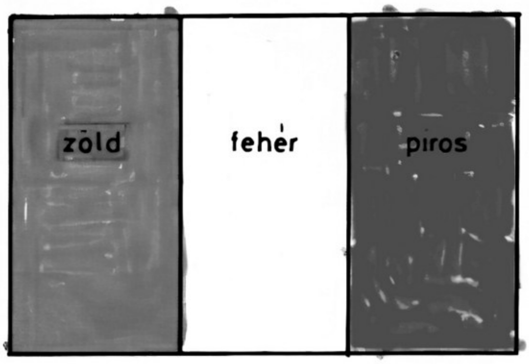
Ha egy zászlón végigvonul egy átlós vonal (mint a tanzániain), jobb, ha a 40 oszlopból csak 25-öt használ és a képernyő többi részét a keretszínnel tölti be. Ha feketét veszünk fel keretnek, a tanzániai zászló közepéig húzódó jellegzetes vonalat így lehetne beindítani:

CTRL és zöld CTRL és RVS ON
SPACE SPACE
CTRL és RVS OFF SHIFT és £ SPACE
CTRL és fekete CTRL és RVS ON
SHIFT és £ SPACE SPACE
CTRL és RVS OFF SHIFT és £ SPACE
CTRL és kék CTRL és RVS ON
SHIFT és £ SPACE ... SPACE

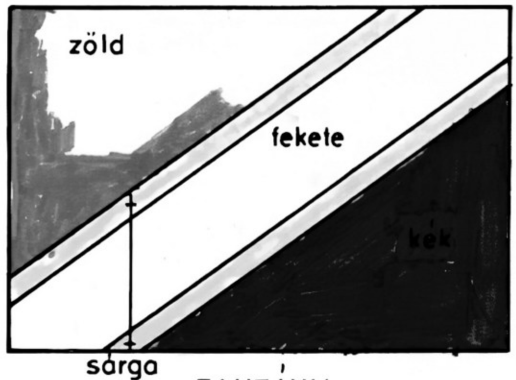
A vesszőket és az 'és' szót csak azért írtuk be az előbbi ábrába, hogy könnyebb legyen átlátni a különböző parancsokat. Ezeket természetesen nem kell használni a zászló megrajzolásakor.

Kísérletezzen a következő ábra zászlóival. A brit zászlót rendkívül nehéz megrajzolni (még számítógéppel is), úgyhogy azt nyugodtan kihagyhatjuk a gyakorlásból.

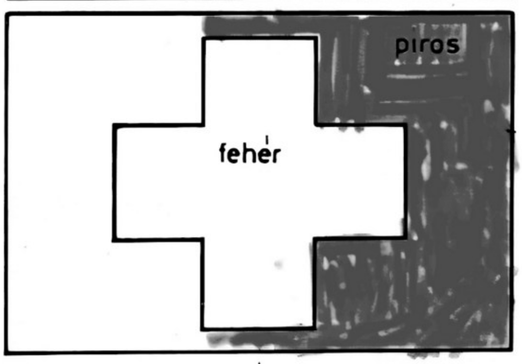
Vége a 3.3 Gyakorlatnak



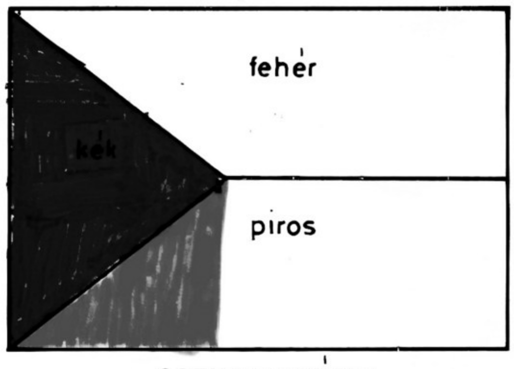
OLASZ



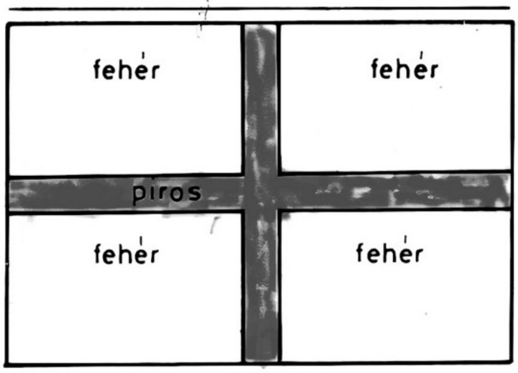
TANZÁNIA



SVÁJC



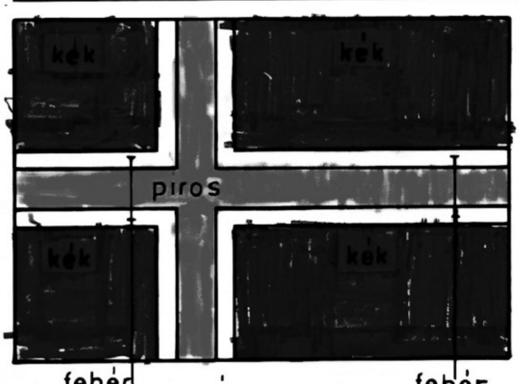
CSEHSZLOVÁKIA



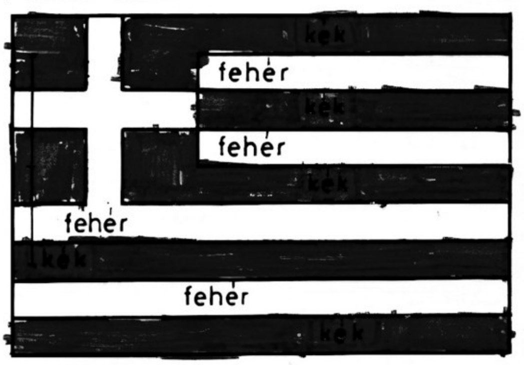
Szt. GYÖRGY



Szt. ANDRÁS



ÍR



GÖRÖG

3.4 Gyakorlat

A COMMODORE 16 és Plus/4 még egy hasznos adottsága az, hogy villogó betűi is vannak. Bármelyik karaktert lehet villogtatni, majd a villogást leállítani (ugyanolyan ütemben villognak, mint a kurzor). A villogást két billentyű vezérli, a „Flash on" (villogás indul) és a „Flash off" (villogás leáll), amelyek a szóköz billentyű közelében találhatók.

Használatuk nagyon hasonlít a RVS ON és RVS OFF használatához. Írjuk le az alábbiakat pontról pontra, hogy egy példán lássuk:

SZAVAZZON Flash on ÉS CTRL

FLASH GORDONRA Flash off ÉS

CTRL ELNOKRE

Használja az egész képernyőt, és rajzoljon rá egy olyan színes képet, amilyet csak tud. Jól teszi, ha előre megtervezi a képet egy kockás papírlapon színes filctollakkal.

Vége a 3.4 Gyakorlatnak

A 3. leckéhez tartozó program egy kérdés-felelet játék, és az alábbiak leírásával lehet betölteni:

LOAD"LECKE3KERDESEK"

vagy

OPEN 1,8,15 "I"

LOAD"LECKE3KERDESEK",8

4. LECKE

4.1 GYAKORLAT

4.2 GYAKORLAT

A könyv első három leckéjében szinte kizárólag csak a számítógép billentyűzetére koncentráltunk, és arra, hogy hogyan használjuk a szövegek megjelenítésére és rajzok készítésére. Ez volt a felkészülés a továbbiakra, ahol megismerhetünk néhányat azokból a tevékenységekből, amelyeket a számítógép elvégez számunkra.

Mint már tudja, a számítógép különböző munkákat képes elvégezni, ha arra parancsot kap. A szükséges parancsokat BASIC nyelven kell megadni, ezen az egyszerű és közérthető számítógépes nyelven, amelyet Kemeny és Kürtz terveztek meg először a Dartmouth College-ban (USA). A BASIC nyelvnek ugyanolyan nyelvtani szabályai vannak, mint bármelyik más nyelvnek, de örömmel bocsátjuk előre, hogy ezek a szabályok egyszerűek, a gyakorlat során könnyen, minden erőfeszítés nélkül megjegyezhetők.

Minden BASIC parancs egy kulcsszóval kezdődik, mint pl. LOAD vagy POKE vagy PRINT. Ebből tudja meg a gép, hogy milyen típusú parancsról van szó.

Hasonló módon, minden parancsa a RETURN billentyűvel végződik. Két különböző jelentése van. Ha például leírja, hogy

LOAD "TESZTKARTYA"

a tényleges betöltés akkor kezdődik meg, amikor a RETURN billentyűt lenyomja.

A RETURN másik értelmezését a következő leckében tárgyaljuk, de már itt is megemlítjük:

Ha a BASIC parancsban a kulcsszó előtt áll egy szám is, pl.: 3 PRINT 5 + 7

akkor RETURN billentyű azt jelzi a gépnek, hogy ne hajtsa végre a parancsot, hanem tárolja későbbi használatra. Ilyen esetben 'parancs' helyett 'utasításról' beszélünk.

Ebben a leckében az első értelmezéssel foglalkozunk. A parancsaink előtt nem lesznek számok, és RETURN közli a számítógéppel, hogy azonnal kezdjen el cselekedni.

4.1 Gyakorlat

Az egyik leghasznosabb és legrugalmasabb parancs a PRINT. Erre a számítógép kidolgoz valamit, és kiírja az eredményét a képernyőre. Azért a PRINT [PRINT = nyomtass *(a lektor megjegyzése).*] szót használjuk, mert Darmouth-ban az eredeti BASIC rendszer mechanikus távgépírókon alapult, amelyek papírtekercsekre nyomtatták ki az eredményt.

A 4.1 Gyakorlat három lépésből áll. Először kipróbálunk egy pár különböző PRINT parancsot, és gondosan lejegyezzük az eredményeket. A második lépésben tárgyaljuk a parancsnak azokat a tulajdonságait, amelyek a példákban felmerültek, és végül megvizsgálunk néhány új PRINT parancsot, és megpróbáljuk előre kitalálni, hogy a számítógép mit fog kezdeni velük. A válaszokat magával a számítógéppel lehet ellenőrizni.

Kezdje leírni az alábbi parancsokat úgy, hogy mindegyik után lenyomja

a RETURN billentyűt. Ellenőrizze a leírt parancsok helyességét, és ha szükséges, az íráshibák javítására használja a kurzor vezérlő billentyűket. Jegyezze le gondosan a válaszokat az alábbi üres keretbe. Az első két keretet mi már kitöltöttük:

PRINT 999	999 READY.
PRINT "HELLO"	HELLO READY
PRINT – 56	
PRINT 3+4+4	
PRINT 5*7	
PRINT 27/7	
PRINT "COMPUTER"	
PRINT COMMODORE	
PRINT 3,5	
PRINT 3;5	
PRINT "NYUL", "KUTYA"	
PRINT "MACSKA", "HAL"	
PRINT "3+5"	
PRINT 29–12; "VADAK"	
PRINT 1;2;3;4;	

Mielőtt tovább olvasna, nézze át újra feljegyzéseit a beírt eredményekről, és nézze meg, hány különböző tulajdonsága van a PRINT-nek. A válaszoknak egyik közös vonása, hogy mindegyik végén ott áll a READY, de ez minden olyan parancs után ott áll, amelyet közvetlenül hajtunk végre, így hát aligha kezelhetjük a PRINT sajátos tulajdonságaként.

Íme a PRINT parancs legfontosabb tulajdonságai:

1. A számokat és a füzéreket [Füzér v. sztring, pontosabban karakterfüzér: karaktersorozat, szöveg *(a lektor megjegyzése).*] egyaránt kezelni tudja, mindegyiket különböző módokon:

A számokat vagy önmagukban adjuk meg (mint a 999-et), vagy egy kifejezés vagy „összeg" formájában, amelyet a számítógép kiszámol. Mintáinkban a kifejezések a következők voltak: 3+4+5, 5*7, 27/7 és 29–12, amiből látható, hogy a számítógép tud összeadni (+ jellel), szorozni (* jellel), osztani (/ jellel) és kivonni (– jellel). (Ha bonyolultabb matematikai számítások elvégzésére van szüksége, örömmel hallhatja, hogy ezek a kifejezések rendkívül bonyolultak is lehetnek. Szerepelhetnek bennük zárójelek és mindazok a speciális függvények, amelyek elvárhatók egy tudományos kalkulátortól. Javasoljuk, hogy nézze meg az A Függeléket, amelyet a matematikai érdeklődésű használóknak állítottunk össze.)

A *füzér* dupla idézőjelek közé zárt karakterek *tetszőleges* sorozata. A PRINT parancs egyszerűen csak visszaírja a füzért pontosan úgy, ahogy megadtuk, anélkül, hogy megpróbálná valamiképpen is feldolgozni. A mintákban ezek voltak a füzérek:

"HELLO", "COMPUTER", "NYUL",
"KUTYA", "MACSKA", "HAL",
"3+5" és "VADAK".

Fontos, hogy a "3+5" *nem* kifejezés, még ha annak néz is ki; idézőjelben áll, ezért csakis füzér lehet.

Ha meg akar jeleníteni egy füzért, de elfelejti idézőjelek közé tenni, valószínűleg Ø-t fog kapni (bár néha mást is kaphat).

2. A PRINT parancs két vagy több mennyiséget, illetve füzért is tud egyidejűleg kezelni. Ha a kettő között vessző van,

a második eredményt jóval messzebb írja ki. Ha pontosvesszőt használunk, kevésbé különíti el az eredményeket. Nevezetesen a füzéreket egyáltalán nem választja el (így kaptuk a "MACSKAHAL"-at). A képernyőre kiírt számok azért vannak egymástól elválasztva, mert mindegyik szám előtt mindig áll egy szóköz (vagy egy – jel, ha a szám negatív), utánuk pedig szintén van egy szóköz. Ha a feljegyzései nem tükrözik ezt világosan, ismételje meg a parancsot:

```
PRINT 1;2;3;4
```


és 'mérje' le az eredményt úgy, hogy végigmozgatja rajta a kurzort.

3. Nézzük meg a szóközöket magán a parancson belül. A PRINT kulcsszónak tömörnek kell lennie (tehát betűit nem választhatják el szóközök), és azok a szóközök, amelyek a füzéren *belül* vannak, a füzérhez tartoznak, és a gép lemásolja őket. Egyébként a füzérek és számok közötti szóközöket a gép nem veszi tudomásul, tehát

```
PRINT     3          +5; 7;          8
```

ugyanazt az eredményt adja, mint a

```
PRINT3+5; 7; 8
```

Ez a szabály – miszerint a parancsokban lévő szóközöket a gép mindenütt figyelmen kívül hagyja, kivéve a kulcsszavakban és a füzéreken belül – általánosságban érvényes az egész BASIC nyelvre.

4. Ha magában a kulcsszóban vét hibát, vagy értelmetlen kifejezést ad meg a gépnek (mint például 276/4), a számítógép elutasítja a parancsot ezzel a megjegyzéssel:

```
?SYNTAX ERROR
```

A számítógép ezzel a 'szintaktikus hiba' megjegyzéssel adja tudtunkra, hogy megszegtük a BASIC nyelv szabályait. Ilyenkor ki kell javítani a parancsot, és újra próbálni.

Nézze át a PRINT parancsok itt következő listáját, és próbálja meg kitalálni, mit fog a számítógép velük csinálni. Azt is jelölje, hol lesznek szóközök és hány az eredményben; ez legalább olyan fontos, mint az eredmény kitalálása. Azzal viszont ne töltse az időt, hogy minden alkalommal leírja a READY-t.

Néhány parancsban szándékosan elhelyezett hibák fordul*hat*nak elő.

Ellenőrizze a válaszait a számítógép segítségével. Ha valahol hibát követett el, de nem tudja, miért, menjen vissza az elejére és ismételje meg az egész gyakorlatot, amíg a működés elvei teljesen világossá nem válnak.

Figyeljen fel arra, hogy a szorzást és osztást tartalmazó számításokat a számítógép mindig az összeadás és a kivonás előtt végzi el, mint például:

PRINT 5+2*7 eredménye 19

és

PRINT 5+2+6/3−3 eredménye 6.

Vége a 4.1. Gyakorlatnak

PARANCS	AZ EREDMÉNY ÖN SZERINT	AZ EREDMÉNY A SZÁMÍTÓGÉP SZERINT
PRINT 94		
PRINT 8 – 5		
PRINT 3 * 2+5		
PRINT "AZ ALMA"		
PRINT MOST		
PRINT "1*/3"		
PRINT 3;47		
PRINT 2+2;2 – 2;2*2/2		
PRINT "BABA" ; "RUHA"		
PRIN8 – 7		
PRINT "18", "EGEREK"		
PRINT 53+*7		
PRINT –1; –2; –3		

4.2 Gyakorlat

Ha a számítógép egyszerre csak egy parancsot tudna teljesíteni, nem volna túl nagy értéke számunkra. Legfeljebb annyi, mint egy (nem programozható) kalkulátornak.
A leghasznosabb munkát a számítógép parancsok egész sorozatából álló, ún. „program" végrehajtásával végzi. Minthogy a parancsok sorozatának végrehajtásáról van szó, kell lennie egy olyan eljárásnak, amellyel „rögzítjük az eredményt", azaz megjegyezzük, hol tart a feladat elvégzése, és az egyik végrehajtott parancs eredményét átadjuk a következő parancsnak. Az a memória, amely a parancsok összekapcsolására szolgál, a *változók*ban testesül meg.

Mielőtt megvizsgálnánk a BASIC változókat, lássunk egy analóg példát. Tegyük fel, hogy Ön jegyzi a gólokat egy futballmeccsen.

Mielőtt a meccs elkezdődik, rajzoljon két rubrikát, írja fel a csapatok nevét, és mindkét rubrikába írjon nullát.

Ø	Ø
VASUTAS KLUB	EGYETEMI KLUB

Amikor valamelyik csapat gólt rúg, a megfelelő rubrikában mindig helyettesítse a legutóbb felírt számot ugyanezen szám *eggyel megnövelt* értékével. Törölje ki (vagy húzza át) a régi számot.

A meccs közben ilyen lehet az állás:

Ø ~~1~~ 2	Ø ~~1~~ ~~2~~ ~~3~~ ~~4~~ 5
VASUTAS KLUB	EGYETEMI KLUB

Amikor lefújják a mérkőzést, írja ki az eredményjelző táblára a csapatok nevét a (legutolsó) számokkal együtt

VASUTAS KLUB	2
EGYETEMI KLUB	5

Ebben a példában a rubrikák *változók,* a bennük lévő számok arra szolgálnak, hogy a játék pillanatnyi eredményét mutassák, és szükség szerint változzanak az eredmény alakulása szerint. De a *feliratok* a meccs ideje alatt változatlanok maradnak. A használati utasításuk roppant egyszerű, de ugyanakkor megbízható.

A számítógép memóriája (amelyben 6Ø671 vagy 12277 byte van – emlékszik még?) némileg olyan, mint egy nagy fekete tábla. Amikor a gépet először bekapcsoljuk, a tábla tökéletesen tisztára van törölve. Aztán amikor először említünk meg egy változót, a számítógép „rajzol egy dobozt", amivel a memória egy részét félreteszi, és azzal a felirattal látja el, amit az alkalmazó ember adott neki. Aztán „számot ír a dobozba", így tárolja a megfelelő értéket a korábban félretett memóriában.

Az a BASIC parancs, amely mindezt elvégezteti a géppel, a LET kulcsszót kapta. Vizsgáljunk meg egy ilyen parancsot részletesen:

```
LET X = 5
```

Itt a változó neve X. A számítógép létrehoz egy X nevű dobozt (ha ilyet eddig még nem csinált), és beleteszi az 5-ös számot. Ha már létezik egy X változó, az 5 egyszerűen a korábbi érték helyére lép. Figyelje meg a következő esetet:

	X nem létezik (1. eset)	X már létezik (2. eset)
Elötte	(A memória üres)	[37] X
Utána	[5] X	[5] X

A LET X=5 eredménye

Amikor a LET parancsot adja, a számítógép csak annyit ír ki:

```
READY.
```

A képernyőn semmi jele nincs annak, hogy a gép bármit csinált volna. Szerencsére a PRINT utasítás segítségünkre van, és kiírja a változó értékét, amikor név szerint említjük. Próbálja ki a következő parancs sorozatot:

	Az Ön eredménye
LET Z=14 PRINT Z	
LET Z=31 PRINT Z	

Ha betartja a helyes sorrendet, a Z első kiírt értéke 14, a második 31 lesz. Az első LET parancs egyrészt megalkot egy Z nevű változót, másrészt a 14 értéket adja neki; a második LET pusztán csak megváltoztatja az értéket 31-re.
Ezen a ponton meg kell tanulnunk néhány egyszerű szabályt, amelyek a változókra és neveikre vonatkoznak. A BASIC-ben kétféle változó van:
– numerikus változók, számok tárolására;
– füzér (sztring) változók, füzérek (pl. szavak vagy mondatok) tárolására.
A változók nevének igen szűk a választéka. A numerikus változókat el lehet nevezni egy betűvel, egy betűvel és egy számmal, vagy két betűvel. Íme néhány lehetséges numerikus változó név:

A, X, Z, B5, TX, PQ

A füzér változók neve mindig a $ jellel végződik, egyébként ugyanazokat a szabályokat kell alkalmazni, mint a numerikus változóknál. Például:

C$, Z$, PZ$, DB$

A füzér változók használatának bemutatására írja le:

```
LET T$ = „JO"

PRINT T$; „REGGELT"
```

A LET parancsban az = jelet követő értékeknek nem kell feltétlenül egyszerű számnak vagy füzérnek lennie; lehet egy kifejezés is, sőt a nevükre való utalással a változók aktuális értékét is használhatja. Nézze meg például ezeket az egymást követő parancsokat:

```
LET Q = 5

LET S = Q+3
```

Az első parancs egy Q nevű változót hoz létre (amely a neve miatt numerikus változó) és az értékét 5-re állítja be. A második parancs egy S nevű változót alkot. Aztán veszi a Q értékét, hozzáad 3-t, és az eredményt beteszi S-be.
A működési elv illusztrálására futtassa le ezt a két parancsot, és vizsgálja meg a következő beírásával:

```
PRINT Q; S
```

Most nézze meg a következő sorozatokat és *mondja meg előre* minden PRINT utasítás eredményét:

LET AA = 15 LET B = 33−AA PRINT AA,B	☐
LET D = 3 LET E = D*D+7 LET F = E−D PRINT F; E; D	☐
LET F = 4 LET F = F+1 PRINT F	☐

Sikerült az utolsót is jól eltalálnia? Ezt néhányan egy kicsit furfangosnak találhatják. Korlátlan azoknak a különböző értékeknek a száma, amelyet egy változó tartalmazhat, de egyidejűleg csak egyet. Egy ilyen parancs

```
LET F = F+1
```

azt jelenti: értékeld ki *először* a kifejezést (alapul véve az F értékét és hozzáadva 1-et), azután tedd az eredményt az F rekeszbe, ezzel helyettesítve a korábbi értékeket. Más szóval erre a parancsra a számítógép *hozzáad 1-et* az F aktuális értékéhez.

Azokat a jeleket, amelyekkel a számokat különböző kapcsolatokba tudjuk hozni egymással, *aritmetikai operátoroknak* (matematikai műveleti jeleknek) hívják. Ezek a +, −, * és /. A BASIC azt is lehetővé teszi, hogy a füzéreket különböző módon manipuláljuk a *füzér operátorokkal.* Közülük csak egy van, amely összekapcsol két füzért, ezt *„konkatenáció"*-nak, láncolásnak nevezik

és a + jellel írják le. Az operátor egyszerűen hozzákapcsolja a második füzért az első füzér végéhez, így

"HAL"+"PIAC" = "HALPIAC"

Nézze meg az alábbi parancs sorozatot, és mondja meg előre, mi lesz a PRINT eredménye. Azután próbálja ki a számítógépen. Ne felejtse el a szóközt betenni az utolsó idézőjel elé:

LET B$ = "A KUTYA SPACE "

LET C$ = "MEGHARAPJA SPACE "

LET D$ = "ESZTERT SPACE "

LET E$ = B$+C$+D$

LET F$ = D$+C$+B$

PRINT E$ [ ]

PRINT F$ [ ]

A PRINT és a LET két leggyakrabban használt utasítás a BASIC-ben. Érdemes megjegyezni, hogy a számítógép használatánál a PRINT szót egyetlen jellel is lehet helyettesíteni: a kérdőjellel (?). A LET-et pedig teljesen el lehet hagyni. Ez is egy érvényes parancs sorozat

A = 5

B = 17

? A, B

A 4. leckéhez tartozó programot úgy terveztük meg, hogy számos lehetőséget adjon a PRINT és a LET utasítás gyakorlására. A neve LECKE4GYAKORLAT.

Ha úgy érzi, hogy tökéletesen megértette a szám és füzér változók használatát, leállíthatja a programot.

Vége a 4.2. Gyakorlatnak

5. LECKE

5.1 Gyakorlat

Most már itt az ideje, hogy megtudjuk, hogyan lehet a parancsokat tárolni. Kezdjük azzal, hogy megmutatjuk: a számítógép valóban képes parancsok – ekkor már utasítások – elraktározására a memóriában, majd később azok elővételére.
Kapcsolja be a gépet, vagy ha már futtat egy másik programot, mint például a GYORSGEPELES, állítsa le a RUN STOP billentyűvel, és írja be a NEW (új) parancsot (amelyet szokás szerint a RETURN billentyű lenyomása követ). Ennek a parancsnak a hatására a számítógép kitörli a memóriáját éppúgy, ahogy az iskolai táblát tisztára törlik a következő óra előtt. Nem látjuk, hogy a READY válaszon kívül bármi történne, hiszen a memória a számítógép belsejében van.

Most írja le ezt a *sorszámozott* parancsot, azaz *utasítást:*

10 PRINT 13+59

(Fontos: Nem I és O, hanem "egy és nulla"!), majd nyomja le a RETURN billentyűt. Az lesz az egyetlen látható eredmény, hogy a gép elmozgatja a kurzort a következő sor elejére. A 13+59 (A 13+59-nek semmi jelentősége nincs, bármilyen más PRINT utasítás megfelelt volna példának.) eredményét *nem* számítja ki és nem írja ki a képernyőre, ehelyett valami láthatatlan dolog történt: a számítógép megjegyezte az utasítást, és félretette azt a memóriájába.
Ahhoz, hogy ezt ellenőrizzük, töröljük le először a képernyőt

(SHIFT és CLR HOME billentyűkkel), aztán írjuk be ezt a parancsot:

```
LIST
```

(Jelentése: listáz.) Ha mindent helyesen csináltunk, az utasítás másolata újra megjelenik a képernyőn. Ez annak a bizonyítéka, hogy a parancs mindvégig bent volt a gépben. Más módon is megadhatjuk a LIST utasítást: ha lenyomva tartjuk a SHIFT billentyűt, miközben leütjük a hosszú HELP/F7 gombot.
Az eddigiekben a gép tárolta és visszaadta a PRINT utasítást, de nem engedelmeskedett neki. A számítógép még mindig nem mondta meg, hogy mennyi 13+59! Ezért írjuk le:

```
GOTO1Ø
```

(GOTO = menj – eredetileg GO TO.)

Nem elfelejtve, hogy két O betűt használjunk a GOTO-ban, nem nullát.

Ez arra utasítja a számítógépet, hogy *hajtsa végre* a „1Ø"-zel jelzett parancsot. A gép végrehajtja, és a válasz végül megjelenik. Az utasítás nem megy „tönkre", ha kilistáztatjuk vagy elvégeztetjük.

A számítógép egyszerre több utasításra is tud emlékezni. (Ennek határa a memória nagyságától függ, ui. az utasítások, a sorszámok, valamint egyebek tárolásához memóriahely szükséges.) Minden utasítás előtt ott kell állnia a sorszámnak, és minden sorszámnak másnak kell lennie.

A gép mindig a sorszámok növekvő sorrendjében tárolja és listázza az utasításokat, és ebben a sorrendben is engedelmeskedik, hacsak másféle utasítást nem kap.

Gépeljük le:

```
NEW
1Ø PRINT "ELSO SOR"
2Ø A=5
3Ø B=1Ø
4Ø PRINT A;B;A+B
5Ø STOP
```

Ne felejtse el, hogy minden parancs végén le kell nyomni a RETURN billentyűt.

Most próbáljuk meg a LIST-et, aztán a GOTO 1Ø-et, és ellenőrizzük, hogy az eredmény azonos-e azzal, amit vártunk.
A STOP leállítja a program végrehajtását, és kiíratja a géppel a READY üzenetet, ha elérte az utasítás-sorozat végét.

A gép akkor is helyes sorrendben tárolja az utasításokat tartalmazó sorokat, ha mi a sorszámaiktól eltérő rendben gépeljük be őket. Ha például azt gépelte korábban, hogy:

```
3Ø B=1Ø
1Ø PRINT "ELSO SOR"
4Ø PRINT A;B;A+B
5Ø STOP
2Ø A=5
```

ezt az öt parancsot mégis a 1Ø, 2Ø, 3Ø, 4Ø, 5Ø sorrendjében listázza és hajtja végre. Törölje le a képernyőt, majd írja be a NEW parancsot, és próbálja ki.

Eleinte mindig tízesével emelkedő számokat használjon. Ha később úgy dönt, hogy beszúr még egy-két külön utasítást a többi közé, ez az eljárás megkönnyíti a dolgot: ilyenkor használhat köztes sorszámokat, mint pl. 15 vagy 38.

Egyáltalán miért töltjük az időt az utasítások tárolásával? Ennek két fontos oka van:

– Azokat az utasításokat sokkal gyorsabban hajtja végre a számítógép, amelyeket a belső memóriájából vesz elő, mint azokat a parancsokat, amelyeket beírunk.

– A gép többször tud újra és újra engedelmeskedni azoknak az utasításoknak, amelyeket egyszer begépeltünk. Gyakorlatilag minden olyan hasznos munka, amit a számítógép elvégez ismétlés, ezért magától értetődően érdemes a gép memóriájában elhelyezni az utasításokat, ahonnan könnyen és gyorsan előhívhatók. Talán úgy érhető el legkönnyebben az ismétlés, ha *tárolunk* egy GOTO *utasítást.* Gondoljuk át a következő programot:

```
1Ø PRINT "ESZAK"
2Ø PRINT "NYUGAT"
3Ø PRINT "DEL"
4Ø PRINT "KELET"
5Ø GOTO 1Ø
```

Ha a 1Ø sorszámmal indítjuk, a gép az első négy parancsnak sorrendben engedelmeskedik. A következő parancs visszaküldi a 1Ø-es sorszámhoz, így aztán újból kezdi a sorozatot. Így megy körbe-körbe, és csak akkor áll le, amikor lenyomja a RUN STOP billentyűt vagy kikapcsolja a gépet.

Most törölje le a képernyőt, és írja be a programot. Indítsa el a

GOTO 1Ø

kezdeti paranccsal.

Látni fogja, hogy a gép olyan gyorsan engedelmeskedik a program sorainak, hogy el sem lehet olvasni. A C= gomb lenyomásával le lehet lassítani a gépet, vagy a szokásos módon le lehet állítani a RUN STOP billentyűvel.

Ezen a ponton megmutatjuk, hogy milyen tényleges előnye van a 1Ø-esével számozott sorszámok használatának. Tegyük fel, hogy meg akarja változtatni a programot a köztes égtájak bevonásával:

ÉSZAK

ÉSZAK-NYUGAT

NYUGAT

DÉL-NYUGAT

stb.

Új utasításokra van szükség a már meglevők *között.* Ha 15, 25, 35, 45 sorszámmal látjuk el őket, pontosan be fognak illeszkedni a megfelelő helyre. Írja le az alábbiakat:

```
15 PRINT "ESZAK-NYUGAT"

25 PRINT "DEL-NYUGAT"

35 PRINT "DEL-KELET"

45 PRINT "ESZAK-KELET"
```

Listázza ki a programot, és ellenőrizze, hogy az új sorok bekerültek-e a régiek közé, a megfelelő helyekre. Futtassa is le a programot, és figyelje meg, mi történik.

Írjon és tanulmányozzon egy hasonló elveken alapuló saját programot. Ha a füzérekben grafikus jeleket is használ egy-egy betű helyett, érdekes minták alakulhatnak ki a képernyőn.

A GOTO 1Ø parancsnak, amelyet eddig a program elindítására használtunk, van egy kényelmesebb megfelelője: a RUN. [RUN = fuss (ha a program éppen működik, azt mondjuk, hogy „fut") *(A lektor megjegyzése).*] A RUN parancsra a számítógép elkezd engedelmeskedni a memóriájában tárolt utasításoknak, a legkisebb sorszámtól kezdve.

Ha a PRINT utasítást követő utolsó füzér *után* pontosvesszőt teszünk, a számítógép nem kezd új sort a következő PRINT utasítás füzéreinek, hanem közvetlenül az előző füzér után folytatja a kiírást. (Ettől függetlenül, minden egyes parancs után le kell nyomni a RETURN billentyűt.) A gép csak akkor kezd el egy új sort, amikor elérte a képernyő jobb szélét. Egy olyan egyszerű program, mint amilyet itt bemutatunk, gyorsan betölti az egész képernyőt különös motívumokkal; próbálja ki és magyarázza meg, hogyan működik:*

1Ø PRINT " 

2Ø GOTO 1Ø

Vége az 5.1 Gyakorlatnak

* A füzér a következőképpen állítható elő: C= és Q, C= és W, C= és R, C= és E, SHIFT és P, SHIFT és L (a lektor megjegyzése.)

5.2 Gyakorlat

Az újra és újra ismétlődő utasítások sorozatát ciklusnak nevezzük. A ciklus több különböző fajta utasítást tartalmazhat, köztük a LET-et, amely új értéket ad a változónak. Nézze meg ezt a programot, és próbálja előre megmondani, mi lesz az eredménye.

```
10 LET A = 1

20 PRINT A

30 LET A = A + 1

40 GOTO 20
```

Tegyen úgy, mintha Ön lenne a számítógép, és csinálja meg ugyanazt, amit a számítógép csinál, türelmesen, lépésről lépésre haladva. Írja le, mi történik az A változóval és értékeivel. Ne olvasson tovább, amíg nem törte a fejét eleget és nem írta be a válaszait.

(és így tovább)

Most írja be a programot, és futtassa le, közben lenyomva tartva a C= billentyűt, hogy lelassítsa a gépet. (De ne érjen hozzá a C=-hez, mielőtt lenyomta volna a RETURN-t a RUN leírása után.)

Biztos vagyok benne, hogy a feladatot könnyűnek találta, és most következik a magyarázata annak, amit látott. A program azzal kezdődik, hogy engedelmeskedik a 10 sorszámú utasításnak, amely a változónak az 1 értéket adja. A következő utasítás kiírja ezt az értéket a képernyőre.

A '30' utasítás A-t A+1-gyel helyettesíti be. Ez ugyanaz, mintha 1-et hozzáadna a régi értékhez. Az eredmény (a mi példánkban) 2. A következő utasítás a GOTO, amitől a gép visszatér a 20 utasításhoz! A számítógép újra kidolgozza a 20, 30, 40 sorszámú utasítást, aztán újra és újra, de minden körben 1-gyel növeli A értékét. Így létrejön az 1,2,3... sorozat.

Gyakorlatképpen próbálja előre megmondani, melyek lesznek ennek a két programnak az első kiírt sorai. (Ne feledje: a * azt jelenti 'szorozva'):

10 B = 0 20 PRINT B 30 B = B+3 40 GOTO 20	10 A = 1 20 B = A * A 30 PRINT A, B 40 A = A+1 50 GOTO 20

És most ellenőrizze, hogy helyesen gondolkodott-e.

Ugyanezt meg lehet tenni a füzérekkel is. Próbálja meg ezt a programot:

```
10 X$ = "*"

20 PRINT X$

30 X$ = X$ + "#"

40 GOTO 20
```

Ahogy a program körbemegy a cikluson, az X$ egymást követő értékei *,*#,*##,*###, stb. lesznek. Az X$ füzér egyre hosszabb lesz, és minden kiírásnál egyre több helyet foglal el a képernyőn. Körülbelül 45 másodperc után a füzér olyan hosszú lesz, hogy nem fér el a számára kijelölt helyen (egy füzér legnagyobb megengedett karakterszáma 255), és a gép hibát jelez:

?STRING TOO LONG

(a füzér túl hosszú)

ERROR IN 30

(hiba a 30 sorban)

Az ERROR IN 30 sor azt jelenti, hogy 30-as volt a sorszáma annak az utasításnak, amelyik megpróbálta tárolni a vétkes füzért.

Bemutatunk még néhány programot, amelyeknek az eredményét előre megmondhatja:

10 A$ = "++" 20 PRINT A$ 30 A$="A"+A$+"–" 40 GOTO 20	10 A$ = "XY" 20 PRINT A$ 30 A$ = A$+A$ 40 GOTO 20

Ne feledje, hogy ha egy betű kerül a füzér *belsejébe,* az csak betűnek számít, és nem egy változó névnek. Tehát az „X"-nek semmi köze az X vagy X$ változókhoz.

Utolsó gyakorlatként írjon egy programot egy egyszerű ciklussal, és pontosan egy percig futtassa (mérje az időt). Aztán állítsa le, nézze meg, mennyi ment le a programból, és *számítsa ki,* hány parancsot hajtott végre a gép ennyi idő alatt. Számítsa ki a *másodpercenként* végrehajtott utasítások számát, és írja ide a számot:

Vége az 5.2 Gyakorlatnak

Az 5. leckéhez tartozó önfelmérő kérdés-válasz program neve: LECKE5KERDESEK.

6. LECKE

Az egész tanfolyam tulajdonképpeni célja, hogy megtanítsa a használót arra, hogyan tervezze meg és építse fel a saját programjait. A programozásban való egyre nagyobb jártasságához szüksége lesz a technikák vagy „szerszámok" olyan kis gyűjteményére, amelyek megszervezik a munkáját és segítenek az elrontott dolgok helyrehozatalában. Ez a lecke egy szerszám- és javítókészlet. Nem magával a programkészítéssel foglalkozik, de a tartalmára mindenesetre nagy szüksége lesz Olvassa gondosan végig ezt a leckét, ismerkedjen meg a leírt technikákkal, fogásokkal, és tartsa észben az itt leírtakat a tanfolyam egész idejére.

Ha kazettás egységet használ a program betöltésére, ezt a részt átugorhatja.

Ha viszont lemezegységgel dolgozik, szeretnénk, ha egy új hajlékony lemezt megformálna, mielőtt elkezdi a 6.1 gyakorlatot.

Mint tudjuk, egyetlen lemez több különböző program tárolására használható. Minden lemezen van egy további tétel: ez a 'directory', a tartalomjegyzék vagy katalógus, amelyik a lemezen levő programok nevét és terjedelmét tartalmazza.

Próbálja meg elvégezni a következőket: tegye be a lemezmeghajtóba a könyvhöz tartozó programlemezt, és írja le:

DIRECTORY RETURN

Ez a parancs beolvassa a lemezről a tartalomjegyzéket a számítógépbe, és kiírja a képernyőre, körülbelül ezt olvashatjuk:

```
 0 "BEVEZ.C16.BASIC"   NT 2A

17 "TESZTKARTYA"       PRG

27 "AKASZTOTT EMBER"   PRG

16 "GYORSGEPELES"      PRG
```

és így tovább, addig, hogy

442 BLOCK FREE

(blokk szabad)

Érdemes tanulmányozni ezt a listát. A legfelső sor, amely inverzben (sötét háttérrel) jelenik meg, a lemezt azonosítja. Ebben az esetben a BEVEZ.C16.BASIC nevet a forgalmazó cég adta meg. Az "NT" egy azonosító, amely szintén a lemezhez tartozik.

A következőkbem minden programra van egy sor. Az első adat a program terjedelmét adja meg *blokkban.* Minden blokk 256 byte-ot tartalmaz, így például az AKASZTOTT EMBER 27 × 256, azaz 6912 byte hosszú. A második adat a program neve, a harmadik, a PRG pedig minden programnál azonos.

A tartalomjegyzék utolsó sora azt mutatja meg, hogy hány szabad blokk van még a lemezen. A szóban forgó lemezen még 16 AKASZTOTT EMBER nagyságú programnak van helye (ha a program rövidebb, még többnek). Egy teljesen üres lemez befogadóképessége 664 blokk. Egy másik, gyorsabb módja a tartalomjegyzék megjelenítésének az f3 f6 billentyű (önmagában való) lenyomása. Most, hogy már tudja, mi van a programlemezeken, továbbléphetünk.

Vegyen elő egy új, megfelelő típusú üres lemezt. Ellenőrizze, hogy a betöltéskor a lemez jobb oldali nyílása üres, és nincs ráragasztva egy olyan ezüst címke (A címke megakadályozza, hogy a lemezegység valamit is ráírjon a lemezre, és így a tanfolyamhoz tartozó programokat megvédi az esetleges sérüléstől. De mivel most Ön a saját programját fogja rögzíteni a lemezen, nincs szükség rá, hogy megvédje.), mint amilyen a BEVEZ.C16.BASIC lemezen látható. Kapcsolja be a lemezmeghajtót, és helyezze be a lemezt. Aztán írja le:

HEADER "lemez neve", DØ lid RETURN

ahol a „lemez neve" helyén az a név áll, amit Ön kíván adni, mint például MARI PROGRAMJA. A cím nem lehet hosszabb 16 karakternél. A D utáni Ø a lemezegység száma. [Egyetlen lemezegység esetén ez mindig Ø *(a lektor megjegyzése).*] A vessző után áll az azonosító, amelyet egy nagy I betű vezet be. Az azonosító két tetszőleges karakter. Ilyen lehet egy valódi beírandó sor:

HEADER "MARI PROGRAMJA", DØ IØ1

A RETURN lenyomása után a következő íródik ki:

ARE YOU SURE?
(biztos benne?)

Ugyanis, ha olyan lemezt formálnánk, amelyen már vannak programok, azok mind elvesznének. Ha biztosak vagyunk a dolgunkban, válaszoljunk Y-nal vagy YES-szel (igen) és természetesen RETURN -nel.

Várjon kb. 1 percig, míg a lemezegység piros lámpája ki nem alszik. Ekkor írja le:

DIRECTORY RETURN

A képernyőn látható egy *üres* tartalomjegyzék, amin csak ez van:

Ø "MARI PROGRAMJA" Ø 1 2A

664 BLOCKS FREE

Ezzel megformálta a lemezt. Minden újonnan vásárolt lemezt meg kell formálni.

Vegye ki a megformált lemezt a meghajtóból, írja rá a címét a címkére és tegye el.

6.1 Gyakorlat

Töltse be és futtassa le a MONDATOK című programot, amely a 6. leckéhez tartozik. Nézze meg a találomra összeállított mondatokat a képernyőn. Ezek az abszurd megállapítások a programban tárolt belső szabályszerűségek alapján álltak össze, miközben minden szó vagy szóegyüttes véletlenszerűen választódott ki a lehetséges szavak egy rövid listájából. Most ne törődjünk azzal, hogy a program hogyan működik (bár az alapelv igen egyszerű), hiszen csak példaként használjuk annak bemutatására, hogy hogyan kell a hosszabb programokat listázni, megváltoztatni és megőrizni.

Ha már eleget nézte a mondatokat, állítsa le a programot a RUN STOP billentyűvel és listáztassa ki. [A programot kisbetű/nagybetű módban célszerű listázni az ékezetes betűk miatt *(a lektor megjegyzése)*.] A program túlságosan hosszú ahhoz, hogy elférjen a képernyőn, és ahogy a listázás fut, a keret tetejénél nagy része eltűnik. Végül csak az utolsó tizenegy utasítást lehet látni.

Erre az esetre a LIST parancsnak van néhány speciális változata a BASIC nyelvben. Öt lehetőség van, próbálja ki mindegyiket:

– Kilistáztathatja az egész programot a LIST legépelésével vagy a SHIFT és f7 billentyű lenyomásával. Ennek, mint már láttuk, hosszabb programok esetén hátrányai vannak.

– Ki lehet listáztatni az utasítások egy részét, ha megadjuk a számukat. Például:

LIST 11ØØ

az 11ØØ-es utasítást fogja kiírni (és mást nem).

– Egy bizonyos sorszám*ig* minden utasítást ki lehet listáztatni, ha a szám elé – jelet írunk. Így:

LIST –8Ø

a program első utasításától a 8Ø-as sorszámú utasításig mindent ki fogja írni.

– Egy bizonyos sorszám*tól* a program végéig minden utasítást kikérhetünk, ha – jelet írunk a szám *után:*

LIST 9Ø9Ø–

– És végül, a program bármelyik (két sorszám *közötti*) részét kiírathatjuk, ha a LIST parancsban megadjuk a kezdő- és végsorszámot:

```
LIST 2ØØØ–2Ø9Ø
```

Most ezekkel a LIST parancsokkal keresse ki a program különböző részeit. Hamar rá fog jönni, hogy a sorszámok nem mindig tízesével növekednek, mivel az első megírás után többször változtattunk a programon.
A programok elején és egyéb helyeken is találkozik majd olyan utasításokkal, amelyekben a REM kulcsszó szerepel, s ezt magyar nyelvű magyarázó mondatok követik. A REM a 'remark' angol szó rövidítése, jelentése 'megjegyzés'. A REM sorok a programban semmilyen szerepet nem játszanak, azért vannak mégis benne, hogy könnyebb legyen elolvasni a programot. Amikor majd bonyolultabb programokat ír, használjon sok REM-et, amivel megmagyarázza, hogy mit csinál programjának egy-egy része.

Amikor úgy érzi, hogy tökéletesen megértette a LIST parancs különböző formáit, próbáljon ki néhány kísérletet, figyelje meg, mi történik, ha:

a) Nincs olyan sorszámú utasítás, amire Ön utal (próbálja meg a LIST 65Ø-et)

b) A sorszámok rossz sorrendben vannak a parancsban (például LIST 11ØØ–1ØØØ)

c) A LIST parancsot utasításként használja a programban. Írja le NEW, aztán az alábbi programot, és futtassa le.

```
1Ø PRINT "LIST PROBA"

2Ø LIST

3Ø GOTO 1Ø
```

Nyomja le a C= billentyűt, hogy pontosan lássa, mi történik. Végül állítsa le a programot a RUN STOP billentyűvel.

Vége a 6.1 Gyakorlatnak

6.2 Gyakorlat

Ez a gyakorlat azt mutatja be, hogyan lehet a programokat megváltoztatni és módosítani. Egyelőre egy olyan programban fog változtatásokat végrehajtani, amelyet eredetileg másvalaki írt, de később a legtöbb módosítást a saját programján fogja végezni.

Egy programot háromféleképpen lehet megváltoztatni:
a) a meglévő utasítások törlésével,
b) új utasítások hozzáadásával,
c) a meglévő utasítások módosításával vagy kicserélésével.

Meglévő utasítások törlése

A számítógépben tárolt, sorszámozott parancsokat 8 módon lehet törölni, de közülük 5 a program teljes kitörlésével vagy megváltoztatásával jár, és túl mélyreható változást eredményez.
Az egész programot ki lehet törölni:
– a gép kikapcsolásával;

– a NEW parancs beírásával;
– új program betöltésével kazettáról vagy lemezről;
– a RESET gomb lenyomásával;

– a SYS 32768 RETURN parancs beírásával.

Egy egyedi utasítást ki lehet törölni:
– sorszámának egyedüli beírásával;
– egy másik utasítás beírásával ugyanazon sorszámmal.

Több utasítást együtt lehet törölni, ha leírjuk, hogy DELETE (törölni), és az első és utolsó sor számát, amelyeket törölni akarunk. Például:

```
DELETE 15Ø–23Ø
```

Kitörli a 15Ø-es és a 23Ø-as sort, valamint az összes köztük levő sort is.

A DELETE parancsnak majdnem ugyanolyan változatai vannak, mint a LIST-nek:

DELETE – 1ØØ	minden sort kitöröl 1ØØ-ig.
DELETE 12Ø–	minden sort töröl 12Ø-tól a program végéig.

DELETE 2ØØ csak a 2ØØ. sort fogja törölni.

DELETE önmagában nem törli ki az egész programot; beírására az a kijelzés, hogy SYNTAX ERROR.

Ha használja a DELETE parancsot, legyen nagyon óvatos. Egyetlen hibával hatalmas kárt tehet a programban! Töltse be újra a MONDATOK programot, és töröljön néhány REM kulcsszóval jelzett sort. Úgy ellenőrizze a törlések helyességét, hogy kilistáztatja a program megfelelő részét a törlés előtt is és után is.

Új utasítások hozzáadása

Egy új utasítással úgy lehet bővíteni a programot, ha azt begépeljük egy megfelelő sorszámmal. Az új utasítás a sorszám által meghatározott helyre fog bekerülni. Az 5. leckében már gyakoroltuk az utasítások beszúrását, de most is megragadhatja az alkalmat néhány REM utasítás beszúrására. Vigyázzon rá, nehogy egy meglevő sor helyére írja be, mert akkor a program nem fog működni.

A meglévő utasítások megváltoztatása

A legdrasztikusabban úgy lehet megváltoztatni egy utasítást, ha átgépeljük ugyanazzal a sorszámmal.

Próbáljunk ki egy változtatást. Kezdjük azzal, hogy a szerzői jogot tartalmazó sort (5 sorszám) helyettesítjük egy sorral, amely az Ön nevét tartalmazza. A párbeszéd így fog kinézni:

```
                   →LIST 5
                    5 REM COPYRIGHT
                    ©MOLNÁR ÉVA 1984
Ezt Ön gépeli be  →5 REM BARNA ANTAL
                   →LIST5
                    5 REM BARNA ANTAL
```

Próbáljon megváltoztatni még egy pár sort, de csak olyanokat, amelyekben a REM kulcsszó áll, mert különben nagyon valószínű, hogy elrontja a programot. A program olyan, mint egy élő sejt: a véletlenszerű mutációk majdnem mindig ártalmasak és általában végzetesek is.

Ha egy sorban csak egy kis változtatásra van szükség, egyszerűbb az eredetit megváltoztatni (ami már a képernyőn van), mint beírni egy új változatot. Ezt a kurzorral és az INST DEL billentyűvel végezzük. Amikor a változtatást befejeztük, a RETURN billentyű megnyomására az új utasítás kerül a régi helyére.

Tegyük fel, hogy meg akarja változtatni a 1ØØ-as sort úgy, hogy így nézzen ki:

100 REM FOE BUTA MONDATGENERATOR

Listáztassa ki a 1ØØ-as sort, vigye a kurzort a MONDAT M betűje fölé, és szúrjon be 5 üres helyet (a SHIFT és INST DEL billentyűvel).

Aztán írja le a BUTA szót a végén egy szóközzel, ellenőrizze, hogy minden rendben van-e, és nyomja le a RETURN-t. Ellenőrzésképpen listáztassa ki újra a 1ØØ-as sort. Próbáljon ki egy pár ehhez hasonló változtatást, de mindig REM sorokban. Ne feledje, hogy ha nem nyomja le a RETURN-t azután, hogy a kurzorral megváltoztatta a sort, a gép nem fogja észlelni a változtatásokat!

Adjon RUN parancsot a gépnek. Ha a program már nem működik, egész biztosan hibázott a szerkesztésben, például kitörölt vagy megváltoztatott egy utasítást anélkül, hogy észrevette volna. Nem baj, ez elég gyakran megtörténik. Töltse be újra a programot a kazettáról vagy a lemezről.

Bizonyára észrevette, hogy a MONDATOK program jól ismert emberekre vonatkozó kijelentéseket is tett. A lehetséges választék listája elég rövid: a 9Ø7Ø (férfiak) és 91ØØ (nők) sorokban van.

E gyakorlat utolsó részében arra kérjük fel, hogy változtassa meg a listákat úgy, hogy a program az Ön családjára és barátaira vonatkozó mondatokat írjon az eddigiek helyett.

Mind a 9Ø7Ø, mind a 91ØØ számú sornak a DATA a kulcsszava (data = adat). Ezt követi a vesszőkkel elválasztott, idézőjelek közé tett [Találkozni fog idézőjelbe nem tett szavakkal is a további listákban. Bonyolultabb annak ismertetése, mikor hagyható el az idézőjel (lásd Commodore 16 Felhasználói kézikönyvben a DATA leírásánál), viszont soha nem baj, ha van. Ezért ne hagyjuk el. *(A lektor megjegyzése).*] nevek listája. Az utolsó név után egy vessző és a Z betű áll. (A DATA utasítás végén a Z betű használata csak ennek a programnak a sajátossága, nem általában a BASIC nyelvnek. A legtöbb más programban nincs szükség a Z-re a DATA utasítás végén.) A név annyi lehet, ahányat

csak akar. Ha a nevek 2 sornál többet foglalnak el, írjon be egy második DATA utasítást (eggyel magasabb sorszámmal, mint az első). Harmadik és negyedik utasítást is lehet használni. Minden csoportban csak az *utolsó* DATA utasítás végére kell kiírni a Z-t.

Íme néhány lehetőség a 9070 és a 9100 sorokra: [Ne felejtsük el, hogy a magyar változat az ékezetes betűk miatt kisbetű/nagybetű üzemmódban működik *(a lektor megjegyzése).*]

```
9070 data "gabi", "gyuri", "Kovács úr",
     "Szofoklész", "Az igazgató úr", Z,

9100 data "Mami", "Zsuzsi", "Viola",
     "A Pillangókisasszony",
     "A tornatanárnő", "Gabi néni", "Kati"

9101 data "Sarolta", "Anna", "Laura",
     "Franciska", "Nóra", "Viki", Z
```

A magyar változatban a társhatározó ragja miatt ugyanazon nevek listája megismétlődik a 9370–9371, illetve a 9200–9205 sorokban a -val, -vel raggal ellátva. Ezért ezeket is át kell írni:

9370 data "Gabival", "Gyurival",......... stb.

Amikor már minden változtatást végrehajtott, futtassa le újra a programot. Hogyha csupa zagyvaságot ír ki, ellenőrizze, hogy tett-e vesszőt (de csak egyet!) minden szó közé, minden szót idézőjelbe tett-e, és ott állt-e az utolsó név után a Z.

Ha úgy érzi, hogy kellő gyakorlatra tett szert a változtatásban, használja szabadon a fantáziáját a programban szereplő további szavak listáival. Ezek a következők (a lista első sorát adjuk meg, addig tart, amíg egy DATA sor végén nincs Z):

9000 Olyan cselekvések, amelyeket az emberek maguk végeznek (tárgyatlan igék)
9010 Olyan cselekvések, amelyeket emberek egymással végeznek (tárgyatlan igék)
9020 Ruházattal kapcsolatos cselekvések (tárgyas igék)
9030 Olyan ruhadarabok, amelyeket férfiak viselnek (tárgyesetben)
9040 Olyan ruhadarabok, amelyeket nők viselnek (tárgyesetben)
9050 Olyan határozók vagy határozószók, amelyek az emberek egymással végzett cselekvésére vonatkoznak
9060 Olyan határozók vagy határozószók, amelyek az emberek maguk által végzett cselekvésére vonatkoznak
9070 Férfinevek
9080 Férfiakra vonatkozó jelzők (névelővel)
9090 Különböző férfiak (foglalkozások, névelővel)
9100 Női nevek
9110 Nőkre vonatkozó jelzők (névelővel)
9120 Különböző nők (foglalkozások, névelővel)
9200 Ugyanazon női nevek, mint 9100-ban, de -val, -vel raggal
9220 Ugyanazon női foglalkozások, mint 9120-ban, de -val, -vel raggal
9370 Ugyanazon férfinevek, mint 9070-ben, de -val, -vel raggal
9390 Ugyanazon férfi foglalkozások, mint 9090-ben, de -val, -vel raggal

Alakítsa át ezeket a listákat kedve szerint, de legyen következetes. Ha a 9020 sort átalakítja ételekre vonatkozó cselekvésekre, a 9030 és a 9040 sort is át kell alakítania, ha nem akar ilyen mondatokat létrehozni:

Zsuzsi megette a gumicsizmáját.

Vége a 6.2 Gyakorlatnak

6.3 Gyakorlat

Elképzelhető, hogy szeretnénk megtartani azokat az új, szellemes mondatokat, amelyeket a MONDATOK program átalakításával hoztunk létre. Jó lenne majd másokat is szórakoztatni velük. Ez a részt azt mondja el, hogyan lehet a programokat a magnókazettán megőrizni. Akinek lemezegysége van, ugorja át ezt a részt, s helyette olvassa el a következőt.

Vegyen egy üres kazettát, vagy olyat, amely nem üres ugyan, de letörölhető. Legyen jó minőségű és minél rövidebb; mivel egy percnél nem akar többet felhasználni belőle, kár kidobni a pénzt hosszabb kazettára. A számítástechnikai célra gyártott speciális kazetták a legjobbak (C5, C10, C30). Tegye be a MONDATOK-at tartalmazó kazetta helyébe az újat, és tekerje a szalagot a legelejére. Ellenőrizze, hogy a magnón egyetlen gomb se legyen lenyomva. Állítsa le a MONDATOK programot, és írja be:

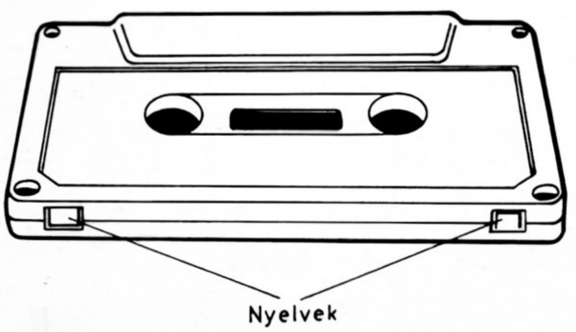
SAVE* "CSALAD" RETURN

(CSALAD helyett bármilyen más név használható, 16 karakterig.)
Erre a gép azt válaszolja:

PRESS RECORD AND PLAY ON TAPE

(Nyomd meg a kazettás egységen a 'felvétel' és 'lejátszás' gombot.)

Kövesse a gép utasítását, és nyomja meg egyszerre a két gombot. Ha a 'felvétel' gombot nem lehet lenyomni, ellenőrizze, hogy rajta vannak-e a kazettán az 'írást megengedő' nyelvek, amelyek a kazetta hátulján találhatók:



* SAVE = mentsd

Ha a lapkák letörtek, semmilyen új anyagot nem lehet már a kazettára felvenni. A kazettát azért alakították ki így, hogy megvédjék a fontos, letörölhetetlen felvételeket, így tehát venni kell egy másikat. [Jó megoldás a nyílás leragasztása műanyag szalagragasztóval (pl. TIXO) *(a lektor megjegyzése)*.] Ha minden rendben van, a gép azt írja ki:

SAVING CSALAD,

és egy pillanat múlva ezt:

READY.

Elvben már a kazettán van a program, de nem árt ellenőrizni. Több hiba csúszhatott a folyamatba; lehet, hogy elfelejtette visszatenni a szalagot az elejére, vagy nem nyomta meg a 'felvétel' gombot, vagy a kazettán is lehetett egy kis hibás rész, amitől megintcsak lehetetlen pontosan rögzíteni a programot. Ezeknek a dolgoknak persze nem kellene megtörténniük, de a gyakorlatban mégis megtörténnek!
Kezdjük az ellenőrzést azzal, hogy a kazettát visszatekerjük az elejére, aztán gépeljük be:

VERIFY* "CSALAD" RETURN

A gép válasza ez:

PRESS PLAY ON TAPE

(Nyomd meg a kazettás egységen a 'lejátszás' gombot.)

Most csak a PLAY ('lejátszás') gombot nyomjuk le. Ekkor a gép megkeresi a kazettán a programot, és összeveti azzal, ami a memóriában van. Természetesen semmilyen változtatást nem szabad végezni rajta a SAVE és a VERIFY utasítások kiadása között.
Ha rendben van minden, ezeket az üzeneteket írja ki a gép:

VERIFY "CSALAD"

PRESS PLAY ON TAPE

SEARCHING FOR CSALAD

(Keresem...)

FOUND CSALAD

(Megtaláltam...)

VERIFYING

* VERIFY = ellenőrizd

(Ellenőrzöm...)

OK

(Rendben)

Ha a gép hibát talál, a FOUND CSALAD kiírásáig sem jut el. Ilyenkor vissza kell menni az elejére, és újra kezdeni az egészet a SAVE utasítástól. Ha a hiba ettől sem szűnik meg, másik kazettával kell próbálkozni (vagy a kazetta másik oldalával). Ha így sem működik a rendszer, akkor géphiba van, és meg kell javíttatni. Ha a programot kimentettük, lehet tárolni és bármikor be lehet olvasni a következő parancs segítségével:

LOAD "CSALAD"

Ha lemezre akarja menteni a programot, vegye ki az BEVEZ.C16.BASIC lemezt, és tegye be azt a lemezt, amelyeket a lecke elején megformált.
Rögzítse a saját programját:

DSAVE "programnev" RETURN

ahol azt írja be a „programnev" helyére, amit Ön választott. Például:

DSAVE "csalad" RETURN

Ha jó a parancs, a gép ezt válaszolja:

SAVINGØ:CSALAD

egy pillanat múlva pedig ezt:

READY.

Ha meg akar győződni róla, hogy a programot hibátlanul rögzítette, írja be:

VERIFY "programnév", 8 RETURN

Biztosan észrevette, hogy míg a DSAVE után nem állt szám, a VERIFY utasítást egy vessző (,) és a 8-as szám követi. A gép válasza ez:

SEARCHING FOR programnév

VERIFYING

OK

READY

Ez a folyamat gyakorlatilag mindig így játszódik le, de ha mégsem, ismételje meg nagyon gondosan az egészet az elejétől a végéig.

Ha kimentette a programot (DSAVE), és ellenőrizte (VERIFY), töltse be és listáztassa ki a tartalomjegyzéket. Most már az Ön programjának is benne kell lennie. A DSAVE" sorozatot gyorsabban begépelhetjük, ha

a SHIFT billentyűvel együtt lenyomjuk

a f2 f5 billentyűt. Ha a programot kimentettük, a szokásos módon vissza is lehet tölteni, például a következő parancs segítségével:

DLOAD "CSALAD" RETURN

Most, hogy már megformált egy lemezt, több programot is tárolhat, amíg van hely a lemezen. Lehet, hogy ki akar majd törölni egy programot, hogy a helyére egy másik változatot írjon be ugyanazzal a programnévvel. Ha csak ezt az utasítást adja:

DSAVE "programnév"

és egy ilyen nevű program már van a lemezen, villogni kezd a figyelmeztető piros lámpa, de semmi nem történik. Ha ki akarja *törölni* a régi programot, egy további karakterre van szükség a program neve előtt:

Most így néz ki a parancs:

DSAVE "@: programnév"

Célszerű, ha egy ilyen DSAVE parancsot egy speciális paranccsal folytatunk:

OPEN 1,8,15,"V"

és megvárjuk, amíg a piros fény kialszik. Ezzel lehetőséget adtunk a gépnek arra, hogy összeszedje azt a helyet, ami most a régi program kitörlésével felszabadult, úgy, hogy az újra felhasználható legyen.

Egy programnak nem kell tökéletesnek lennie ahhoz, hogy kimenthessük. Ha nagyon hosszú programot ír (vagy kimásol egy könyvből), érdemes körülbelül félóránként kimenteni az addig leírtakat. A számítógép memóriája ugyanis kevésbé megbízható, mint a tartójában elhelyezett kazetta vagy mágneslemez. Ugyan a számítógép sem romlik el könnyen, de véletlenek mindig közbejöhetnek. Hallottunk már arról, hogy a vihar villámlása megrongálta a számítógépben tárolt információt; lehet hálózati áramkiesés, és az is megtörténhet,

hogy valaki átgázol a hálózati vezetéken és kirántja a csatlakozót a falból. Joggal keseredik el az ember, ha öt-hat órás munkáját egy ilyen apró véletlen teszi tönkre. Ha félóránként rendszeresen elraktározza az anyagot, újra lehet tölteni a legutolsó változatot és csekély időveszteséggel folytatni lehet a munkát.

Akkor lesz a rendszer a legbiztonságosabb, ha felváltva két kazettára vagy két lemezre menti ki a programot. Így, még ha a kimentés közben áll le a gép, s a régi program felét kitörölte az új program fele, mégis van honnan újraindulni.

Vége a 6.3 Gyakorlatnak

6.4 Gyakorlat

Ebben a részben azokra a ravasz és veszélyes csapdákra hívjuk fel a figyelmét, amelyek minden COMMODORE számítógép programozóra leselkednek. Azt is elmondjuk, hogyan lehet kimászni belőlük, ha már beleestünk.

Már a legelején bedobjuk a mélyvízbe: egy egyszerű csapda kellős közepébe. Gépelje be a NEW-t, ezzel kitörli a tárolt információt, majd írja be a következő programot. Írás közben figyelje a képernyőt, és gondosan tartsa be a szóközöket.

```
10 PRINT "EZ EGY NAGYON VESZELYES CSAPDA"
20 GOTO 10
```

Listáztassa ki, és ellenőrizze a programot, aminek pontosan úgy kell kinéznie, mint fent.

Azt várjuk, hogy a RUN lenyomásától a program újra és újra leírja ezt az üzenetet, amíg le nem állítjuk:

EZ EGY NAGYON VESZELYES CSAPDA

Próbálja csak meg! Valószínűleg ez jön ki belőle:

```
EZ EGY NAGYON VESZELYES CSAPDA 2Ø

?SYNTAX ERROR IN 1Ø

READY
```

Ha a programja helyesen működik, akkor is olvassa el az alábbiakat, hogy megtudja, mivel kerülte el a kelepcét.

Távolról sem egyértelmű, hogy miért nem tudta a gép lefuttatni a programot (feltéve, ha erről van szó). A BASIC világhírű szakemberei sem találnak benne hibát.

A gondot az okozza, hogy a számítógép képernyője keskeny. A gépen belül minden BASIC utasítás 75 karakter hosszúságú lehet. A képernyő csak 40 karakter szélességű, tehát az utasítás *kijelzett* változata két sort foglalhat el a képernyőn.

Amikor begépelünk egy utasítást, és a kurzor eléri a képernyőn a sor végét, a rendszer továbbmozgatja a következő sor elejére, de még mindig azt feltételezi, hogy Ön *ugyanazt* az utasítást írja. Egy utasítást csakis a RETURN billentyűvel lehet befejezni.

A következők miatt esett bele a csapdába (feltéve, ha beleesett):

Ön leírta az első utasítást (amit szándékosan úgy terveztünk meg, hogy a képernyő *egész* sorát kitöltse). Ekkor Ön azt látta, hogy a kurzor a következő sor legelején áll, és természetesen leírta a következő parancsot, végén a RETURN -nel. Mivel az első sor végén nem ütötte be a RETURN -t, a gép azt hitte, hogy mind a két sor ugyanannak az utasításnak a része, tehát

```
1Ø PRINT "EZ EGY NAGYON
VESZÉLYES CSAPDA" 2Ø GOTO 1Ø
```

Ez az „utasítás" BASIC nyelven hibás, és a gép szintaktikus hibának tekinti, mikor végre akarja hajtani.

Az ilyen típusú hibát nagyon nehéz megtalálni, hacsak nem tudjuk előre, hogy mit keresünk. Aligha hihető, hogy Ön gépelés közben magától észreveszi ezt a hibát – még a gyakorlott programozók is gyakran elfelejthetik lenyomni a RETURN -t, ha a kurzor egy új sor elején áll. A hibás sor ugyanúgy néz ki, mint a helyes, amikor kilistáztatja a programot, vagy külön azt a részt, amelyikben a hiba van, ezért aztán nem lehet látni.

Szerencsére azonban el lehet kapni a hibát, ha csak azt a sort listáztatja ki, amelyikben a hiba van. Ha beírja: LIST 1Ø, kiírja a 1Ø-es sort és vele együtt megjelenik a 2Ø-as! Itt valami hibának kell lennie, hiszen csak a 1Ø-es sort kértük. A dolgot úgy hozhatjuk helyre, hogy mind a két utasítást újraírjuk végig, most már mindkettő végén a RETURN lenyomásával.

Összefoglalás:

a) Mindig minden utasítást a RETURN -nel fejezzünk be, függetlenül attól, hogy hol áll a kurzor.

b) Ha a számítógép azt jelzi, hogy hiba van az utasításban, és Ön nem látja, hogy hol, listáztassa ki a hibás sort önmagában, és ellenőrizze, hogy nem fut-e át a program következő utasításba.

Vége a 6.4 Gyakorlatnak

7. LECKE

7.1 GYAKORLAT

7.2 GYAKORLAT

7.3 GYAKORLAT

Azok a programok, amelyeket az 5. leckében leírtunk, fegyelmezetlenek voltak. Ha egyszer beindítottuk, csak drasztikusan lehetett őket leállítani, és ha nem avatkoztunk volna be, a végtelenségig mentek volna tovább. Ebben a leckében azt fogjuk megtanulni, hogyan lehet a programokat vezérelni, valamint leállítani, ha már megtették a dolgukat.

Az itt leírtak alapvető jelentőségűek a számítógép megfelelő alkalmazásában. Ezen ismeretek elsajátításával tesszük meg a legnagyobb egyedi lépést a programozóvá válás felé. Olvassa el azt a fejezetet lassan és gondosan, és ha valami nem világos, olvassa el újra elölről. Érdemes ennyi időt töltenie vele, mert ha ezeket az elveket egyszer megérti, nagy előrehaladást könyvelhet el.

A programok vezérlésének kulcsa – amely Önnek új lehet – a *feltétel*.

Az emberek közötti beszélgetésekben elhangzó kijelentések többnyire igazak, vagy legalábbis feltételezik, hogy igazak:

„Elromlott az autóm, amikor ide tartottam."

„Szeretlek."

A feltétel olyan sajátos kijelentés, amely nem szükségszerűen igaz; ugyanolyan valószínűséggel hamis is lehet. Angolul az 'if' (ha) szó vezeti be a feltételt. Az alábbi mondatokban a feltételek vastag betűvel vannak szedve:

„Ha **az utolsó vonat is elment,** itt kell töltened az éjszakát."

„Ha **a program nem működik,** keresd meg a hibát és javítsd ki."

Akik ezeket a mondatokat mondták, nem állítják, hogy az utolsó vonat elment, vagy hogy a program tényleg nem működik; éppenhogy nem tudják biztosan, ezért felkészülnek az eshetőségekre. A beszélt nyelvben a feltételről kiderülhet az is, hogy igaz, az is, hogy nem, de ettől még nem hazudott, aki mondta.

A feltételek a BASIC nyelvben is az IF kulcsszó után állnak. Ugyanazokat a dolgokat tartalmazzák, mint a programok: numerikus

változókat, füzér változókat, számokat és füzéreket. A feltételek, amelyek bármelyik példában lehetnek hamisak vagy igazak, 6 reláció köré csoportosíthatók. Ezeket példákkal lehet a legjobban megmagyarázni:

Figyeljük meg a BASIC feltételt:

A<5

(ahol a < jel azt jelenti, hogy „kisebb, mint"). Ez a feltétel akkor igaz, ha az A változó értéke valóban kisebb 5-nél (mondjuk 0 vagy 3, vagy 4.98*). A feltétel *nem igaz,* ha A értéke 5 vagy több.

Egy másik példa, most füzérekkel:

N$<>"ANNA"

(ahol a <> jel azt jelenti, „nem egyenlő"). Ez a feltétel akkor igaz, ha az N$ változó értéke bármi, csak nem „ANNA", tehát igaz, ha N$=„TOMI" vagy N$=„ANNI". A feltétel csak akkor hamis, ha N$ ténylegesen „ANNA".

A BASIC-ben ismert relációkat a következő jelek fejezik ki:

= (egyenlő)
< (kisebb, mint)
> (nagyobb, mint)
< > (nem egyenlő)
< = (kisebb vagy egyenlő)
> = (nagyobb vagy egyenlő)

A < >, < = és > = viszonyokat *két* billentyű leütéssel gépelhetjük be. Lehet, hogy a ≠, ≤ és ≥ jelek ismerősebbek, de a BASIC tervezőinek azt kellett figyelembe venniük, hogy a számítógépek billentyűzetén általában nincsenek meg ezek a jelek.

A *feltételek* létrehozásakor ezeket a relációkat két szám vagy két füzér között lehet alkalmazni. A számokat és a füzéreket megfelelő változók képviselhetik.

5>4 *igaz,* mert 5 nagyobb, mint 4.
7<=6 *hamis,* mert 7 nagyobb, mint 6.
Ha A = 10 és B = 7,
A> = B *igaz,* és B< = 7 is az.

Ha az egymás közti viszonyokat füzérekre alkalmazzuk, az ábécé sorrendje érvényes, tehát:

„ALMA" < „BABA" igaz,
de „BAB" > „BABA" hamis.**

* A BASIC-ben tizedespontot kell használni tizedesvessző helyett *(a lektor megjegyzése).*

** Az ékezetes betűket is tartalmazó füzérekre nem alkalmazhatók a relációk (az ékezetes betűk nem sorrendezhetők) *(a lektor megjegyzése).*

7.1 Gyakorlat

Tegyük fel, hogy a számítógép engedelmeskedett ennek a három utasításnak:

LET A$ = "GIZI"

LET X = 5

LET Y = 7

Ennek alapján írja be az alábbi táblázatba, hogy melyik feltétel igaz, és melyik hamis:

Feltétel	Érték (igaz vagy hamis)
X<7 X>=5 A$<>"X" Y<>X A$>"ZSUZSA" A$<"GIZI" Y=8	

A relációjelek mindkét oldalán álló mennyiségek lehetnek kifejezések, éppúgy, mint a LET utasításokban. A kifejezések bármilyen összetettek lehetnek, az egyetlen fontos dolog az, hogy azonos típusú dolgokat állítsunk relációba: az olyan feltételtől, amelynek az egyik oldalán szám van, a másikon pedig füzér, a számítógép leáll, és hibát jelez.

Dolgozza fel az alábbi feltételeket, feltételezve, hogy A$, X és Y értéke ugyanaz marad, mint az előbbi példában.

Feltétel	Érték (igaz vagy hamis)
A$+"KE"<>"GIZIKE" 5>X X+Y<>13 X+2=Y	

És most ellenőrizze a válaszok helyességét a B Függelékben.

Vége a 7.1 Gyakorlatnak

7.2 Gyakorlat

A BASIC nyelvben a vezérlés legfontosabb eszköze az IF ('ha') utasítás. Ez a következőkből tevődik össze: Az IF kulcsszóból, egy feltételből, a THEN ('akkor') kulcsszóból és egy sorszámból. Nagyon hasonlít a GOTO utasításhoz, de van egy lényeges különbség: itt ugrás csak akkor lesz, ha a feltétel *igaz*.

Egy példa az IF utasításra:

```
IF X$<>"ABBBB" THEN 2Ø
```

Itt a feltétel az X$<>"ABBBB", az egész utasítás arra utasítja a gépet, hogy a 2Ø-as sorra ugorjon, ha X$ nem egyenlő "ABBBB"-vel. Ha a feltétel hamis, a gép folytatja az utasítások végrehajtását számozásuk növekvő sorrendjében.

Valószínűleg Ön is azt fogja először gondolni, mint a legtöbb ember, hogy ez egy kicsit képtelen dolog. „Az X$ vagy nem egyenlő az A-kból és B-kből álló füzérrel, vagy egyenlő" – gondolja önmagában. Minden attól függ, hogy mi van előbb, mindenesetre a programozó tudhatta, mikor megírta az IF utasítást!

Az elképzelése érthető, elfogadható, de hamis. Két teljesen különböző ok miatt:

– Tételezze fel, hogy az IF utasítás olyan ciklusban van, ahol valamelyik változó értéke minden alkalommal megváltozik. A feltétel igaz lehet néhány értékre, de a többire nem.

– Tételezze fel azt is, hogy valaki másnak készít egy programot. Nem tudhatja előre a használó szándékát, de a program mentésének mégis attól kell függnie, hogy mit csinál valójában a használó. Egy jó példa: e könyv szerzőjének úgy kellett kérdés-felelet játékokat összeállítani, hogy érzékenyen reagáljon az Ön válaszaira, holott fogalma sem lehetett róla, hogy mik lesznek majd ezek.

Ha egy ciklus már elégszer körbement, könnyen le tudjuk állítani a bennefoglalt IF utasítással. Gépelje be és futtassa le a következőt:

```
1Ø X$ = "A"

2Ø PRINT X$

3Ø X$ = X$+"B"

4Ø GOTO 2Ø

5Ø STOP
```

A program továbbfut, és megtölti a képernyőt B-ből álló, egyre hosszabb füzérekkel, amíg a füzér hossza el nem éri a 256-ot. Sohasem éri el a STOP-ot az 5Ø. sorban, mivel hibaüzenettel áll le a fenti eset elérésekor.

Helyettesítse a 4Ø. sort ezzel:

```
4Ø IF X$<>"ABBBB" THEN 2Ø
```

Amikor lefuttatja, a gép azt jelzi ki:

```
A

AB

ABB

ABBB, és megáll!
```

Ennek oka az X$<>"ABBBB" feltétel. Ahogy a program körbehalad a cikluson, a feltétel először igaz (mert X$ értéke az AB, aztán ABB, majd ABBB, és ez mind egyenlő ABBBB-vel). Ezekben az esetekben az IF utasítás mindig úgy viselkedik, mint egy egyszerű GOTO, és még egyszer körbeküldi a gépet a cikluson. Végül az X$ eljut az ABBBB-hez. Itt már hamis a feltétel, a visszalépés nem történik meg, a gép ráfut a program végére, ahol megáll.

Most próbálja különbözőképpen megváltoztatni a feltételt, és figyelje meg, mi lesz az eredménye, amikor lefuttatja a programot. Bármilyen füzért használ, arra vigyázzon, hogy a feltétel végül hamis legyen, mert különben a program sohasem áll le.

Ki lehet próbálni ezeket a lehetséges feltételeket:

```
X$<>"AB"
X$<>"ABBBBBBBBBB"
X$<>"ABBA"
```

Ugyanezt a vezérlési technikát alkalmazhatja számváltozókkal is. Írja be:

```
1Ø P=Ø

2Ø PRINT P,P*P

3Ø P=P+1

4Ø GOTO 2Ø

5Ø STOP
```

Futtassa le ezt a programot, nézze meg, hogy működik, állítsa le, és változtassa meg a 4Ø. sort erre:

```
4Ø IF P<11 THEN 2Ø
```

Most futtassa le újra a programot. Az két számoszlopot fog kijelezni, amelyek nagyon ismerősek, és hasznosak lehetnek azoknak, akik nem tudják fejből a számok négyzeteit. Csak egy baja van mint munkaprogramnak: a táblázaton nincs fejléc, ezért a jelentése csak Önnek egyértelmű, másnak nem. Ezt a hiányosságot is rendbehozhatjuk, ha a tetejére írunk egy sort, amely az oszlopokat megnevezi, így:

SZAM	NEGYZET	
Ø	Ø	
1	1	
2	4	
3	9	
...	...	és így tovább.

A címsornak értelemszerűen a számok, ill. négyzeteik előtt kell megjelenni, ezért ezt az utasítást a program elejére kell írni. Mivel a 1Ø sorszámot már lefoglaltuk, és értelmetlen volna az egész program sorszámozását megváltoztatni, ésszerű az 5 sorszámot használni. Maga az utasítás a PRINT, két füzérrel: "SZAM" és "NEGYZET". A füzérek közötti vessző (,) azt teszi lehetővé, hogy a szóközök megfeleljenek a számoszlopok közötti szóközöknek.

Most így néz ki az egész program:

```
 5 PRINT "SZAM", "NEGYZET"

1Ø P=Ø

2Ø PRINT P,P*P

3Ø P=P+1

4Ø IF P<11 THEN 2Ø

5Ø STOP
```

Futtassa le a programot ebben a formában, és vizsgálja meg az eredményt.

Szeretné, hogy a címsort egy üres sor válassza el az első számsortól? A PRINT utasítás önmagában (minden más érték vagy füzér hozzáadása nélkül) létrehoz egy üres sort, tehát próbálja hozzátenni a következő utasítást:

```
7 PRINT
```

Rövidesen az lesz a feladata, hogy írjon saját maga programokat. De mielőtt hozzáfogna, nézzük meg alaposan azokat a programokat, amelyeket már lefuttattunk, és vonjunk le belőlük néhány általános következtetést. Ezek a mintaprogramok:

(1)

```
1Ø X$="A"
2Ø PRINT X$
3Ø X$=X$+"B"
4Ø IF X$<>"ABBBB" THEN 2Ø
5Ø STOP
```

(2)

```
 5 PRINT "SZAM", "NEGYZET"
 7 PRINT
1Ø P=Ø
2Ø PRINT P, P*P
3Ø P=P+1
4Ø IF P<11 THEN 2Ø
5Ø STOP
```

Ha a második programban eltekintünk a címsortól (5 és 7), úgy tűnik, hogy mindkét program azonos felépítést követ. Mindkét esetben:

1. Van egy változó, amely a ciklus ismétlődésével rendszeresen megváltozik. Ez az első programban az X$, a másodikban a P. Általában ezt *ciklusváltozónak* nevezzük, lehet akár füzér, akár szám.

2. Van egy utasítás, amely megadja a ciklusváltozó *kezdőértékét.* Ez a parancs a cikluson kívül van (tehát nem ismétlődik, a gép csak egyszer hajtja végre).

3. Van egy vagy több utasítás, amelyet a gép a ciklusváltozó minden értékére végrehajt. A mi példánkban ezek a PRINT utasítások

PRINT X$

és PRINT P, P*P

A ciklusnak ezt a részét gyakorlatilag meg lehet növelni tetszőleges számú olyan utasítással, amelyeknek *mindegyikét* végrehajtja a gép a ciklusváltozó minden értékére. Ezt a csoportot hívjuk a ciklus *magjának* vagy *törzsének.*

4. Van egy *növekmény* vagy mennyiség, amivel a ciklusváltozó növekszik, valahányszor körbemegy a ciklus. A mi példánkban X$

a "B" hozzáadásával növekszik, P pedig 1-gyel növekszik. Másféle növekedés is elképzelhető, egy füzér például egyszerre 5 karakterrel növekedhet, egy szám növekedhet kettesével vagy bármilyen más számmal. A ciklusváltozó kezdőértéke lehet nagy, és értéke *csökkenhet.* Ezenkívül a ciklus mindig tartalmaz egy olyan utasítást, amely a ciklusváltozót mindig továbblépteti végrehajtása során.

5. A ciklusváltozónak van egy végértéke. Amikor a ciklus ezzel az értékkel hajtódik végre, az ismétlésnek meg kell szűnnie. A ciklusban az IF az utolsó utasítás egy olyan feltétellel, amely *igaz,* ha a ciklusnak még mindig körbe kell mennie, de *hamis,* ha a ciklusváltozó már fölötte van a végértéknek.

Tanulmányozza alaposan az alábbi táblázatban levő programokat, írja be a ciklusváltozó nevét, a kezdőértéket, a végértéket, a növekményt és azt a számot, ahányszor a gép végrehajtja a ciklust. Érdemes lejegyeznie a ciklusváltozó értékét, amint először, másodszor, harmadszor ... halad át a cikluson, és megállapítania, hány értéke van, mielőtt eléri a végső értéket.

A feladat megoldása után hasonlítsa össze válaszait a könyv végén levő eredményekkel (B Függelék).

Vége a 7.2 Gyakorlatnak

	Ciklus-változó	Kezdőérték	Végérték	Növekmény	A ciklusok száma
1Ø X$="A" 2Ø PRINT X$ 3Ø X$=X$+"B" 4Ø IF X$<>"ABBBB" THEN 2Ø 5Ø STOP	X$	"A"	"ABBBB"	"B"	4
1Ø P=Ø 2Ø PRINT P, P*P 3Ø P=P+1 4Ø IF P<11 THEN 2Ø 5Ø STOP	P	Ø	1Ø	+1	11
1Ø Y$="Z" 2Ø PRINT Y$ 3Ø Y$=Y$+"XY" 4Ø IF Y$<>"ZXYXYXY" THEN 2Ø 5Ø STOP					
1Ø R=5 2Ø PRINT R,R/8 3Ø R=R+3 4Ø IF R<17 THEN 2Ø 5Ø STOP					
1Ø C=27 2Ø H=3Ø−C 3Ø PRINT C,H 4Ø C=C−5 5Ø IF C>2 THEN 2Ø 6Ø STOP					

7.3 Gyakorlat

Egy program felépítését úgy kell elkezdeni, hogy először valamennyire megtervezzük, aztán leírjuk papírra az egész programot. Használjunk ceruzát és *radírt!* Vannak, akik egyenesen a számítógép billentyűzetén állítják össze a programot, de ez a módszer csak zseniknek vagy gyengeelméjűeknek való, semmiképpen sem átlagos embereknek.
A magyarázat igen egyszerű: ha terv nélkül fog a munkához, ugyanannyi sikerre számíthat, mint az a kőműves, aki tervrajzok nélkül húzza fel a házat, és menet közben alakítgatja ki építészetileg. Lehet, hogy egy építészeti csodát fog létrehozni, de az még valószínűbb, hogy olyan rozoga kalyibát csinál, amit az első szélvihar elrepíthet.

Amikor egy ciklust tervez a programhoz, előre el kell határozni minden alapvető részét, vagyis a ciklusváltozó nevét és típusát, a kezdő- és végértéket, a növekményt és a ciklus magjának részleteit. Amikor mindezeket végiggondolta, az ismert minta alapján összerakhatja azokat.

Íme egy kidolgozott példa.

Egy font sterling (1 £) a mai árfolyamon 2350 olasz lírát ér. Olyan táblázatot szeretnénk, amely a font–líra egyenértékét adja meg 5 £-tól 75 £-ig, ötösével haladva. Így indul a kijelzés:

```
£    LIRA
 5   11750
10   23500
...    ...   és így tovább.
```

Először gondoljuk végig a ciklust. A ciklusváltozó nyilvánvalóan egy szám lesz, és hívhatjuk FS-nek (a font sterling rövidítéseként, de bármilyen név is megfelel). A kezdőérték 5 lesz, a végérték 75, a növekmény pedig 5. A ciklus tömbjének ki kell írnia az értéket fontban (£), a neki megfelelő értéket pedig lírában, amely 2350-szer több.

Most már le lehet jegyezni a ciklus elemeit:

FS=5	Beállítja a kezdőértéket
PRINT FS,2350*FS	Ciklusmag
FS=FS+5	Növeli FS-t
IF FS<80THEN	Ellenőrzi, hogy túlhaladtunk-e a végső értéken
STOP	Leállítja a programot

A THEN után álló sorszám helyét üresen hagytuk, mert még nem tudjuk, hogy mi lesz.

Mielőtt leírjuk az egész programot, fontoljuk meg, mi legyen a címsor. Ezek lennének a megfelelő utasítások:

PRINT "£","LIRA"

és PRINT — Hogy egy üres sort kapjunk

Most már összerakhatjuk a különböző részeket, és leírhatjuk az egész programot.

```
10 PRINT "£","LIRA"

20 PRINT

30 FS=5

40 PRINT FS,2350*FS

50 FS=FS+5

60 IF FS<80 THEN 40

70 STOP
```

Hadd ismételjem meg újra, még ha unalmasnak tűnik is: *ne* spóroljon a program megtervezésével; *ne* írja be ötletszerűen a programot egyenesen a gépbe. Ha mégis megteszi, Önből soha nem lesz jó programozó!

Most próbálja ki a következő feladatokat:

52

1. Írjon egy programot, amelyik ilyen csillagmintát hoz létre:

```
*
**
***
****
........................
**********-ig
```

2. Írjon egy olyan programot, amely az US dollár ($) és a brit font sterling (£) egyenértékét adja meg 10 és 30 £ között, két font sterlingenként haladva (1 £ = 1,43 $).

3. A Fahrenheit és a Celsius skálák közötti kapcsolatot ez a képlet fejezi ki:

```
F = 1.8*C + 32
```

Írjon egy olyan programot, amely 1 fokonként haladva táblázatba szedi a Fahrenheitben és Celsiusban mért hőmérsékletet 15 °C és 30 °C között.
(Útmutatásul: a program magja legyen

```
F = 1.8*C + 32

PRINT C,F
```

Ez behatárolja az Ön választását a ciklusváltozó nevére vonatkozóan.)

Amikor a programokat megírta és lefuttatta, hasonlítsa össze a megoldásokat a B Függelékben lévőkkel.

Vége a 7.3 Gyakorlatnak

Akik szeretik az iskolai matematikát és jók is benne, néha egészen megzavarodnak a "=" jel BASIC-beli használatától. Ha Önnek ez nem okoz gondot, ugorja át ezt a részt.

A matematikában a "=" jelet egyenlőségeknél használjuk, amikor azt állítjuk, hogy a két különböző kifejezés ugyanazt az értéket képviseli. Az egyenlőség megállapít valamilyen igazságot. Ha például a matektanár azt írja fel a táblára, hogy

"2x+5 = 9"

biztosak lehetünk benne, hogy azzal a sajátos x-szel, amire a tanár gondolt, helyes az állítás. Ha nem így lenne, elképzelheti a következő párbeszédet:

A diák jelentkezik:
Tanár: Tessék!
Diák: x egyenlő kettővel.
Tanár: Nem. A válasz hetvennyolc.
Diák: Micsoda? Ezt nem értem.
Tanár: Hazudtam, amikor azt mondtam 2x+5 = 9!

A BASIC nyelvben a "=" jelet két különböző értelemben használjuk, de egyik sem azonos a matematikai értelmezéssel.

A LET utasításban az egyenlőségjel azt jelenti, hogy „...értéke legyen" vagy „vegye fel... értéket". Arra *utasítja* a gépet, hogy számolja ki a jobb oldali kifejezés értékét, és ezt helyezze el a bal oldali változóban. [Azaz *értéket ad* a bal oldalon lévő változónak, ezért *értékadó utasításnak* nevezzük *(a lektor megjegyzése)*.] Az utasítások nem állítások, ezért értelmetlen volna azon vitatkozni, hogy igazak vagy sem. (Bizonyos összefüggésben hamisak lehetnek, de ez egy más téma.) A baj csak az, hogy ha a LET-et lehagyjuk, az utasítás *úgy néz ki,* mint egy egyenlőség. Pedig nem az. Tehát tisztázzuk:

A BASIC nyelvben az

Y = X+2*

nem azt *közli* a számítógéppel, hogy Y egyenlő X+2-vel, hanem *elrendeli,* hogy a gép *számolja ki* az X+2 értékét, és vigye be az eredményt az Y változóba. Íme néhány megfontolandó megállapítás:

* Amit így mondhatunk:
Y értéke legyen X+2 (értéke) vagy
X vegye fel az X+2 értéket (a lektor megjegyzése).

- Q = Q+5 ez egy értelmes és hasznos BASIC utasítás
- P = Q
és Q = P a hatásuk nem azonos
- X+1 = 5 *nem* szabályszerű BASIC utasítás

Minden esetben érti, hogy miért? Próbálja megmagyarázni a saját szavaival!

A "=" jel másik használata a feltételekre vonatkozik. Nyilván emlékszik még rá, hogy a "=" az értékek közötti hat lehetséges viszonyból az egyiket fejezi ki. Íme két példa a használatára:


```
IF X = Y+2 THEN 100

IF N$ = "IGEN" THEN 150
```

Ezek sem foglalják magukban azt az értelmezést, hogy a feltétel valóban igaz; itt a parancs elrendeli, hogy a gép értékelje ki, *vajon* igaz-e a feltétel, és ha igen, végezzen el bizonyos tevékenységet. A feltételekben a "=" jelnek ugyanolyan logikai érvénye van, mint bármilyen más relációs jelnek, például a < vagy a >. Az a legjobb, ha nem is használjuk az „egyenlő" szót, hanem úgy hívjuk a jelet, hogy „ugyanaz mint".

Összefoglalva:

A BASIC nyelv a "=" jelet olyan utasításokban használja, ahol a jelentése 'legyen', és olyan feltételekben, ahol azt jelenti: 'ugyanaz mint', de amit mond, az nem feltétlenül igaz. Érthető?

A leckéhez tartozó önfelmérő program neve LECKE7KERDESEK.

8. LECKE

8.1 GYAKORLAT

8.2 GYAKORLAT

8.3 GYAKORLAT

Ezen a számítógépes tanfolyamon Ön most eljutott arra a pontra, amikor éppen elkezdett saját programokat készíteni. Az első programok rövidek és egyszerűek lesznek. Biztos, hogy később, amikor még tovább fejleszti a tudását, gyakorlottságát és jártasságát, egyre bonyolultabb és érdekesebb programokat fog tervezni és írni. Ez a táblázat azt mutatja be, hogy körülbelül milyen messzire juthat:

Program	*Utasítások száma*
Az olasz líra átváltása fontra (7. Lecke)	7
A 3. Lecke kvíz programja	kb. 100
Sakkjáték program	kb. 5000
Egy ipari robotot vezérlő program	kb. 25 000
Egy számítógépes repülőtéri jegyfoglaló rendszer programja	kb. 5 000 000

Természetesen azok a programok, amelyek túlhaladják az 5000 körüli utasításszámot, már minden esetben csoportmunka eredményei (egy embertől túl sok időt igényelne megírása), de ezen belül egy-egy egyedi programozónak így is nagy a hatásköre.

Programozás közben gyakran előfordul, hogy az ember elakad. Az a program, amit nagy gonddal megírt és begépelt, egyszerűen nem azt csinálja, amit várt tőle. Ez a fejezet néhány olyan módszert ír le, amellyel meg lehet oldani e nehézségeket. Olvassa el és végezze el a gyakorlatokat; de ne feledje, hogy erre a részre mindig visszatérhet, amikor (nem *ha*) szükség van rá.

Az emberek különbözőképpen reagálnak, amikor először kerülnek szembe programozási nehézségekkel. Vannak, akik dühösek és ingerültek lesznek, mások azonnal feladják és kétségbeesetten kijelentik, hogy a programozás

nem nekik való; megint mások elhitetik magukkal, hogy a program „kilencvenkilenc százalékban jó", és már át is ugranak a következő problémára. Egyik reakcióból sem sül ki sok jól. Az egyetlen megoldás az, hogy megkeressük a hibát és helyrehozzuk. Gondoljon nyugodtan arra, hogy minden programozó elakad néha, még olyanok is, akik 25 éve dolgoznak számítógépen.

A programhibák három csoportra oszthatók. Az első és leggyakoribb általános hiba az, amelyik szintaktikus hibát (SYNTAX ERROR) jelez vissza, amikor a gép megpróbál végrehajtani egy adott utasítást. Ez azt jelenti, hogy az utasítás nem tartja be a BASIC nyelv formai szabályait. Lehet például egy betűhiba a kulcsszóban, vagy túl sok (esetleg túl kevés) a dupla idézőjel. A legtöbb szintaktikus hibát gépelési hiba okozza, és ezek nyilvánvalóak, ha észrevettük, hogy hol vannak.

A második programhiba típus akkor jelentkezik, amikor a gép egy olyan utasítást talál, amelyet nem lehet végrehajtani. Tegyük fel, hogy a gép elérkezett ehhez az utasításhoz:

```
130 GOTO 500
```

de nincs 500-as sorszámú utasítás. Ekkor a gép leáll, és kijelzi a hibaüzenetet:

```
UNDEFINED STATEMENT IN 130
```
(határozatlan utasítás a 130. sorban).

Sajnos a hibaüzenetek nem hétköznapi angolsággal, hanem számítógépes zsargonnal íródnak ki, de teljes magyarázatukat megtalálja a C. Függelékben.

Ha hibaüzenetet kapunk, a HELP (segíts) parancs használható. Ez a parancs villogva kiíratja a géppel a programnak azt a sorát, amely a hibát okozta.

Próbaképpen írja be és futtassa le ezt a programot:

```
10 PRINT 1+2+3 * * 4
```

(ahol a második * szándékos hiba).

A műveleteket a program nem tudja végrehajtani. Amikor megkapja a HELP parancsot, kiírja a teljes sort villogva.

Elő lehet hívni a HELP parancsot a HELP/F7 billentyűvel is, vagy a H-E-L-P betűk begépelésével és a RETURN-nel.

A harmadik hibatípust a legnehezebb megtalálni és rendbehozni. Ilyenkor nincsenek hibaüzenetek; ehelyett a gép egyszerűen kiírja a rossz választ vagy bennragad egy ciklusban anélkül, hogy bármit kiírna. Az első és legkézenfekvőbb tennivaló az, hogy a gépet leállítjuk, kilistáztatjuk a programot, és alaposan átnézzük. Ilyenkor általában sikerül észrevenni a hibát. De tegyük fel, hogy mégsem, mondjuk már percek óta vizsgálgatja külön-külön az utasításokat, mégsem talál semmilyen hibát.

Ekkor már sokkal hatásosabb módszert kell találni a program menetének vizsgálatára. Ezt a módszert 'nyomon követésnek' hívják, és azt jelenti, hogy megpróbálunk a számítógép fejével gondolkodni. Induljunk el a program elejéről, és menjünk végig parancsról parancsra, amíg hirtelen fel nem fedezzük a hiba okát. Nagyon türelmesen és módszeresen kell haladni, és ami a legfontosabb: *ki kell kapcsolni* a saját értelmünket, és úgy kell végrehajtani az utasítások sorát, mint egy buta robotgép, nem lehet „ésszerűen találgatni", általánosítani vagy bármilyen módon lerövidíteni a lépéseket.

Ahhoz, hogy utánozni tudjuk a számítógépet, először alaposan meg kell ismerni a működését. Tegyük fel, hogy egy program futtatása közben, két utasítás között le tudnánk „fagyasztani" a számítógépet, felnyitnánk és belenéznénk. Ezeket találnánk:*

Először is magát a programot, amelyet a memória az eredeti beíráshoz nagyon hasonló formában tárol.

Másodszor azokat a változókat, amelyeket a program eddig felhasznált. Minden változó egy bizonyos helyet foglal el a memóriában, és *értéke* van, amely lehet szám vagy füzér.

Harmadszor azt látnánk, hogy a számítógép megjelöli a helyet, ahol a programban tart. Valahol (valójában a „programmutató" nevű speciális változóban) tárolja annak az utasításnak a sorszámát, amelyet következő lépésként végre kell hajtania.

Most hagyjuk abba a „lefagyasztást" legalább annyi időre, amíg a gép *egyetlen* utasítást végre tud hajtani. Természetesen azt az utasítást fogja választani, amelyre a programmutató emlékszik. Ha az utasítás végrehajtása után ismét bepillantunk a gépbe, bizonyos változásokat tapasztalunk, amelyeket az éppen végrehajtott utasítás eredményez. Íme néhány lehetőség:

* Ha levennénk a számítógép tetejét, valójában csak néhány szilíciumlapkát (chipet) és egyéb alkatrészeket látnánk. A megfelelő elektronikai eszközök viszont tényleg megmutatnák azokat a dolgokat, amikről most beszélünk.

a) A PRINT utasítás hatására valami megjelenik a képernyőn.

b) A LET utasítás egy új változót fog létrehozni, ha szükség van rá, és egy új értéket ad neki.
Mind a PRINT, mind a LET utasítás egyben tovább is mozdítja a programmutatót a sorban következő utasítás felé, tehát amikor a gépet újraindítjuk, „tudja", hogy melyik utasítás végrehajtása következik.

c) A GOTO nem jelez ki semmit, és nem változtatja meg a változókat. Pusztán csak újraállítja a programmutatót, hogy az a GOTO-ban szereplő utasításra mutasson. Például ez a parancs

```
13Ø GOTO 27Ø
```

be fogja tenni a 27Ø-et a programmutatóba.

d) Az IF utasítás hasonlóan működik, azzal az eltéréssel, hogy a gép először a feltételt értékeli ki. Ha a feltétel *igaz,* bekerül a programmutatóba, mint a GOTO-nál. Ha *hamis,* a programmutató egyszerűen tovább növekszik a soron következő utasítás sorszám értékére.

Nézzünk erre egy példát:

```
12Ø IF X = 5 THEN 17Ø

13Ø PRINT "NEM"
```

Ha X-nek valóban 5 az értéke, a feltétel igaz, és a programmutató értéke 17Ø-re változik. Ha viszont X-nek valamilyen más értéke van, a programmutató egyszerűen előretolódik 13Ø-ra.

e) A STOP utasítás kijelzi a program végét azzal, hogy kiírja a BREAK ('törés, megszakadás') üzenetet. Ezen a ponton túl nem érdemes folytatni a programot. [Hacsak nem éppen egy ideiglenes megállítás céljából helyeztük el a programban, hibakeresés végett. E tanfolyam keretében erről nem lesz szó *(a lektor megjegyzése)*.]

Ahhoz, hogy hitelesen tudjuk utánozni a számítógépet, világosan kell ismerni: a programot, a változókat, a kijelzést és a programmutatót. Hasznos módszer, ha egy „program nyomkövető ábrát" használunk, amelyet felrajzolunk egy papírra. Például így:

PROGRAMMUTATÓ 1Ø	
VÁLTOZÓK	
KIJELZÉS	*PROGRAM*
	1Ø A = 5 2Ø PRINT "ALFA" = ";A 3Ø A = A*3 4Ø B = A+37 5Ø PRINT "BETA = ";B 6Ø STOP

A program a jobb oldalon látható, a programmutató kezdőértéke pedig – azaz az első végrehajtó parancs sorszáma – legfelül. Ügyeljünk arra, hogy a program *pontos* másolata legyen annak, amelyikkel gondunk van; ha nem így van, az egész nyomkövetés csak időpazarlás.

Most már kész vagyunk az indulásra. A programmutató 1Ø-et mutat, tehát nézzük meg, hogy mi a 1Ø-es sorszámú utasítás. Ez A = 5, tehát egy LET utasításnak kell lennie. Keressük ki a VÁLTOZÓKnak nevezett dobozból az A-t. Ilyen még nincs, ezért írjunk le egy "A"-t, egy kettőspontot és az 5 értéket. Végül mozgassuk el a programmutatót a soron következő utasításra, és húzzuk át az előző értéket:

PROGRAMMUTATÓ ~~1Ø~~ 2Ø	
VÁLTOZÓK A:5	
KIJELZÉS	*PROGRAM*
	1Ø A = 5 2Ø PRINT "ALFA" = ";A 3Ø A = A*3 4Ø B = B+37 5Ø PRINT "BETA = ";B 6Ø STOP

A következő utasítást is így kell értelmezni. Felejtsük el a program 'célját', és mindent, amit a program feladatáról tudunk, és a 2Ø-as utasítást *csak* azért vegyük sorra, mert a programszámláló azt mutatja. Az utasítás a PRINT, és rájöhetünk, hogy a gép ezt fogja kijelezni: "ALFA = 5". Ezt írja be a KIJELZÉS rovatba, és növelje tovább a programmutatót:

PROGRAMMUTATÓ ~~1Ø~~ ~~2Ø~~ 3Ø	
VÁLTOZÓK A:5	
KIJELZÉS	*PROGRAM*
ALFA = 5	1Ø A = 5 2Ø PRINT "ALFA = ";A 3Ø A = A*3 4Ø B = A+37 5Ø PRINT "BETA = ";B 6Ø STOP

A következő utasítás új értéket ad a már meglévő A változónak. Először számítsuk ki az A*3 kifejezést a régi értékkel (5), jegyezzük le, majd húzzuk át a régi értéket, így:

A:~~5~~ 15

A következő utasítás ismét egy új változót hoz létre. Folytassuk a nyomkövetést, amíg el nem érjük a STOP-ot. Ez a végeredmény:

PROGRAMMUTATÓ ~~1Ø~~ ~~2Ø~~ ~~3Ø~~ ~~4Ø~~ ~~5Ø~~ 6Ø	
VÁLTOZÓK A:~~5~~ 15	
KIJELZÉS	*PROGRAM*
ALFA = 5 BETA = 52 BREAK IN 6Ø READY	1Ø A = 5 2Ø PRINT "ALFA = ";A 3Ø A = A*3 4Ø B = A+37 5Ø PRINT "BETA = ";B 6Ø STOP

A következő példa egy egyszerű ciklust tartalmaz:

```
1Ø P = 1

2Ø PRINT P;P*P*P

3Ø P = P+1

4Ø IF P<4 THEN 2Ø

5Ø STOP
```

A program nyomkövetése a 3Ø sorig egyszerű:

PROGRAMMUTATÓ ~~1Ø~~ ~~2Ø~~ ~~3Ø~~ 4Ø	
VÁLTOZÓK P: ~~1~~ 2	
KIJELZÉS	*PROGRAM*
1 1	1Ø P = 1 2Ø PRINT P;P*P*P 3Ø P = P + 1 4Ø IF P<4 THEN 2Ø 5Ø STOP

A 4Ø sorban IF utasítás van. Tegyünk úgy, mint a számítógép, és értékeljük a P<4 feltételt. Mivel a P jelenlegi értéke 2 (ezt mutatja a VÁLTOZÓK rész), és mivel 2 nyilvánvalóan kisebb, mint 4, a feltétel igaz. Így az egyetlen tennivalónk, hogy a programmutatót 2Ø-ra állítjuk. Ezt kapjuk:

PROGRAMMUTATÓ ~~1Ø~~ ~~2Ø~~ ~~3Ø~~ ~~4Ø~~ 2Ø	
VÁLTOZÓK P: ~~1~~ 2	
KIJELZÉS	*PROGRAM*
1 1	1Ø P = 1 2Ø PRINT P;P*P*P 3Ø P = P + 1 4Ø IF P<4 THEN 2Ø 5Ø STOP

A nyomonkövetés ugyanígy folytatódik, amíg végre el nem érünk a hamis feltételhez, a program pedig a STOP-hoz. Ez a végeredmény:

PROGRAMMUTATÓ ~~1Ø~~ ~~2Ø~~ ~~3Ø~~ ~~4Ø~~ ~~2Ø~~ ~~3Ø~~ ~~4Ø~~ ~~2Ø~~ ~~3Ø~~ ~~4Ø~~ 5Ø	
VÁLTOZÓK P: ~~1~~ ~~2~~ ~~3~~ 4	
KIJELZÉS	*PROGRAM*
1 1 2 8 3 27 BREAK IN 5Ø READY	1Ø P = 1 2Ø PRINT P;P*P*P 3Ø P = P + 1 4Ø IF P<4 THEN 2Ø 5Ø STOP

Gépeljük be a programot, és futtassuk le. Ellenőrizzük, hogy egyeznek-e az eredmények.

Most vegye elő újra azt a programot, amely kiíratja az ALFA-t és BETA-t, és azt is futtassa le. Ilyen kijelzésnek kell megjelenni:

ALFA = 5

BETA = 52

BREAK IN 6Ø

READY

A számítógép rendelkezik azzal a figyelemreméltó képességgel, hogy önmaga is tud nyomkövetést végezni. Próbálja ki, írja be ezt az utasítást:

TRON `RETURN`

(a TROn a 'TRace ON' – nyomozz, haladj a nyomon – rövidítése), és a programot ismét futtassa le. A programmutató értékei szögletes zárójelben megjelennek a képernyőn:

[1Ø] [2Ø] ALFA = 5

[3Ø] [4Ø] [5Ø] BETA = 52

[6Ø]

BREAK IN 6Ø

Ez azt jelenti, hogy a program ilyen sorrendben hajtotta végre az utasításokat: 1Ø 2Ø 3Ø 4Ø 5Ø és 6Ø. Az

ALFA = 5

kiíratást a 2Ø sor hozta létre, mint ahogy gondoltuk is.

A nyomkövetést kikapcsolhatjuk a következő paranccsal:

TROFF `RETURN`

(TRace OF – 'hagyd abba a nyomkövetést' – rövidítése). Most már segítségül hívhatja a nyomkereséshez a TRON és TROFF parancsokat is. Hamarosan megtanulja, hogy óvatosan kell bánni a nyomkövetéssel, mert egy ciklusból álló program általában igen sok utasításnak engedelmeskedik, és a sorok számának listája könnyen eláraszthatja a képernyőt.

8.1 Gyakorlat

Gyakorolja a nyomkövetést az alábbi programokkal. Ceruzával írjon, és legyen kéznél egy radír is, hátha hibázik valahol!

a)

PROGRAMMUTATÓ 1Ø	
VÁLTOZÓK	
KIJELZÉS	*PROGRAM*
	1Ø X = 5 2Ø Y = 7 3Ø Z = X+Y 4Ø W = Y−X 5Ø PRINT X;Y;Z;W 6Ø STOP

b)

PROGRAMMUTATÓ 1Ø	
VÁLTOZÓK	
KIJELZÉS	*PROGRAM*
	1Ø Q = 1 2Ø PRINT "SZERET" 3Ø PRINT "NEM SZERET" 4Ø Q = Q+1 5Ø IF Q<3 THEN 3Ø 6Ø STOP

Ha elvégezte a gyakorlatokat, ellenőrizze a válaszait a számítógépen: futtassa le úgy a programokat, hogy a nyomkövetést bekapcsolja.

Vége a 8.1 Gyakorlatnak

8.2 Gyakorlat

Hogyan tudjuk hibák megtalálására használni a nyomkövetést? Úgy, hogy megpróbálunk egy engedelmes robot és egy értelmes emberi elme állapota között átkapcsolódni. Először robottá válunk, és megpróbáljuk pontosan úgy követni a parancsokat, ahogy a számítógép csinálta volna. Aztán visszaváltozunk emberré, és megkérdezzük: „Ezt vártam?" Ha igen, folytatjuk a nyomkövetést. Ha nem, akkor már megvan a megoldás kulcsa, mitől romlott el a program.

Vegyünk egy egyszerű példát. Tegyük fel, hogy olyan programot írt, aminek a 12-es szorzótáblát kell kiírnia. Ezt a kijelzést várjuk:

```
TIZENKETTES SZORZOTABLA

1*12 = 12

2*12 = 24

3*12 = 36
```

(és így tovább, egészen eddig):

```
12*12 = 144
```

A programnak minden része megvan: a ciklus, a fejlécet kijelző utasítás és a PRINT utasítás, amely a táblázat egyes sorait kiírja. Így néz ki:

```
10  PRINT "TIZENKETTES
    SZORZOTABLA"

20  P=1

30  P=P+1

40  IF P < 13 THEN 30

50  PRINT P; "*12=";P*12

60  STOP
```

A program végeredménye egy kis csalódást okoz. Csak ennyit kapunk:

```
TIZENKETTES SZORZOTABLA

13*12=156

BREAK IN 60

READY
```

és nem azt, amir számítottunk! Lehet, hogy a hiba teljesen szembeötlő, de tegyünk úgy, mintha nem találnánk. Kezdjük el a nyomkövetést, és néhány lépés után ide érkezünk:

PROGRAMMUTATÓ ~~10~~ ~~20~~ ~~30~~ ~~40~~ ~~30~~ ~~40~~ 30	
VÁLTOZÓK P: ~~1~~ ~~2~~ ~~3~~ 4	
KIJELZÉS	*PROGRAM*
TIZENKETTES SZORZOTABLA	10 PRINT "TIZENKETTES SZORZOTABLA" 20 P = 1 30 P = P+1 40 IF P<13 THEN 30 50 PRINT P; "*12="; P*12 60 STOP

és ekkor hirtelen rájövünk, hogy a P értéke megy felfelé 12-ig, de a gép ebből nem jelez ki semmit. Most már világos, hogy a PRINT utasításnak a cikluson belül kell lennie, és nem kívül! A 20 és 30 utasítás között lesz jó helyen. Az IF utasítást is meg kell változtatni úgy, hogy ugorjon a PRINT utasításra vissza. Egy gyors szerkesztés után az eredmény:

```
10  PRINT "TIZENKETTES
    SZORZOTABLA"

20  P=1

24  PRINT P; "*12"; P*12

30  P=P+1

40  IF P < 13 THEN 25

60  STOP
```

A nyomkövetés roppant hasznos módszer, ha van türelmünk ahhoz, hogy a programot lépésről lépésre hajtsuk végre. Ha a program részleteit futólag nézzük át, nagyon valószínű, hogy ugyanabba a hibába esünk,

mint mikor eredetileg megírtuk a programot, és így nem fogjuk megtalálni a hibát.

Kevesebb időnk megy el, ha a Commodore beépített nyomkövető mechanizmusát használjuk, amely kiírja a végrehajtott utasítások sorszámát. De még ekkor is nagyon alaposan kell elemezni a soronkénti listát, mert a TRON sem helyettesítheti a gondolkodást!

Amikor már biztosak vagyunk abban, hogy a program egyes részei jól működnek, a nyomkövetésre csak bizonyos programrészeknél van szükségünk. Ha elkerüljük a felesleges számok áradatát, a TRON-t és a TROFF-ot utasításként használhatjuk magában a programban:

```
1Ø ------
2Ø ------
....
1Ø5 TRON   }
11Ø ----   }  Nyomkövetett
12Ø ----   }  programrész
.....      }
185 TROFF  }
19Ø -----
2ØØ -----
```

Amikor átnézte a gyanús programrészt, könnyen törölheti a TRON és TROFF utasításokat.

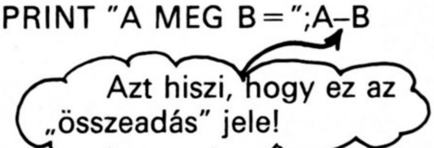
Van néhány olyan körülmény, amelyek között a fent leírt nyomkövetési módszer nem használható. Nevezetesen:

– Ha a program nagyon hosszú, az egyenes nyomkövetés túl sok időt vesz igénybe. Erre vannak jobb módszerek, amelyeket majd akkor mutatunk be, amikor ténylegesen szükség lesz rájuk.

– Ha valaki nem hiszi el saját magáról, hogy hibázik, akkor a nyomkövetéssel nem megy sokra. Sokan ugyanis amikor leírják a program utolsó sorát, határozott biztonsággal úgy érzik: „Ez most már jó!" Ez az érzés azért jön elő, mert nem voltak *tudatában* annak, hogy hibáztak, és ez rendkívül félrevezető. Sokkal jobb, ha azt mondjuk magunknak: „Most biztos rossz. Találjuk meg a hibákat!" Félre kell tenni a büszkeséget!

– Ha valaki félreértette a BASIC nyelv valamely alapvető oldalát, a nyomkövetés újra csak vajmi keveset segít. Hogy egy durva példát vegyünk, tegyük fel, hogy valaki meg van róla győződve – határozottan, de persze hibásan –, hogy a BASIC-ben a "–" jel összeadást jelent. Így akar a programban két számot összeadni:

```
1Ø A=34
2Ø B=19
3Ø PRINT "A=";A
4Ø PRINT "B=";B
5Ø PRINT "A MEG B=";A–B
```



Ha a félreértésen alapuló programot lefuttatja, ezt a kijelzést kapja:

```
A=34
B=19
A MEG B=15
BREAK IN 6Ø
READY
```

ami nyilvánvalóan hibás. Másrészt amikor a nyomkövetést végzi papíron, az 5Ø sorban ezt kapja:

```
A MEG B=53
```

ami pedig megfelel a várakozásának. Soha nem fogja megtalálni a hibát, amíg abban a hitben van, hogy a "–" összeadást jelent!

Természetesen a félreértések nagy része nem olyan durva, mint ez a példa. Ha a nyomkövetés végén más eredményre jutunk, mint a számítógép, és újra megismételt nyomkövetésnél is más eredményt kapunk, akkor biztos, hogy a programkészítés művészetéből nem értettünk meg valamit pontosan. Ilyen esetben lehetőleg forduljunk tanácsért valakihez, aki nálunk jobban ismeri a BASIC-et; másképp menjünk vissza a könyv legelejére, és vessük össze minden ismeretünket a könyvben leírtakkal. Ez végül is majdnem mindig fényt fog deríteni a hibára.

Néha – bár igen ritkán – a számítógép mechanikai hibája okozhatja a bonyodalmat. Az olyan modern gépek, mint a számítógép, rendkívül megbízhatóak, de ha mégis elromlanak, a hiba rögtön láthatóvá válik: bekapcsoláskor a kurzor nem jelenik meg vagy a programokat nem lehet begépelni. De addig ne a számítógépet okoljuk a program hibájáért, amíg nem próbáltunk ki minden egyéb lehetőséget kétszer-háromszor. Amikor elvisszük javíttatni a gépet, meg kell magyaráznunk pontosan, miből következtetünk arra, hogy elromlott, és mellékeljük annak a programnak a másolatát, amit a gép nem tud helyesen lefuttatni.

Most vegye szemügyre az alábbi két programot, és próbálja megkeresni, majd kijavítani a bennük elrejtett hibákat.

a) Ennek a programnak az a feladata, hogy kiírja a gallonból literbe való átszámítás táblázatát 1 gallontól 10 gallonig (1 gallon = 4,5 liter).

```
10 PRINT "GALLON", "LITER"

20 G=1

30 PRINT G, 4.5*G

40 G = G+1

50 IF G >11 THEN 30

60 STOP
```

b) Az alábbi programnak meg kellene oldania a 7. lecke 1 feladatát, azaz ki kellene rajzolnia egy csillagokból álló háromszöget. Olyan valaki írta, aki most tanulja a BASIC nyelvet:

```
10 A$="*"

20 PRINT A$

30 A$="**"

40 IF A$ < > "*********" THEN 20

50 STOP
```

Vége a 8.2 Gyakorlatnak

8.3 Gyakorlat

A LECKE8PROGRAM című program feladata az, hogy kiírja a 7-szeres szorzótáblát, de több hiba van benne. Töltse be a gépbe, keresse meg és javítsa ki a hibákat. Ellenőrizze a válaszai helyességét a B. Függelékben.

Vége a 8.3 Gyakorlatnak

9. LECKE

Rajzoljunk újra képeket! De úgy, hogy a számítógép végezzen el minden nehéz „rabszolgamunkát".

Ha visszagondolunk a 2. és 3. leckére, felidézhetjük, hogy a rajzoláshoz több vezérlő 'funkció' áll rendelkezésünkre:

– a kurzor mozgatása négy különböző irányban,
– színválasztás tizenhat különböző színből,
– a szín és a háttér felcserélése (inverz mód)
– villogó karakterek
– a kurzor 'hazamozgatása' a bal felső sarokba,
– képernyőtörlés.

Ezek a funkciók más célú billentyűket alkalmaznak, ezért a SHIFT vagy a CTRL vagy a  billentyűt gyakran használjuk a kívánt funkció megvalósítására. Reméljük, nem felejtette el, hogy a keret és a háttér színét a 'COLOR' (szín) utasítással és a 24. oldalon található színkódokkal lehet beállítani.

A számítógép programvezérléssel is tud rajzolni a képernyőre. Minden program rendelkezésére áll az összes képernyő vezérlő funkció: bármilyen színt választhat a karakterei kijelzésére; ha szükséges, letörölheti a képernyőt; bármely helyzetbe mozgatni tudja a saját kurzorját (amit mi nem látunk) a kurzor vezérlő funkciók használatával.

Természetesen a számítógép mindezt csak akkor végzi el, ha megfelelő utasításokat kapott tőlünk. A képernyő vezérlő funkciókat könnyű elhelyezni egy utasításban: egyszerűen beírjuk őket a füzérekbe azokkal a karakterekkel együtt, amelyeket ki akarunk iratni. [Ezért szoktuk úgy mondani: a vezérlő funkciót vezérlő karakter valósítja meg *(a lektor megjegyzése)*.] Lehet, hogy ez most kicsit rejtélyesnek tűnik. Tényleg, vajon gépeléskor minden el fog tűnni a képernyőről, ha egy füzérbe begépelünk egy képernyő törlés utasítást? Nem, nem fog eltűnni, mint ahogy ezt a következő gyakorlat is bizonyítja.

9.1. Gyakorlat

Emlékszik még, mit mondtunk a 2. leckében? „Ne használja a dupla idézőjelet, mert nagyon fura dolgokat kaphat!" Most megnézzük, mi is a hatásuk, és miért olyan hasznosak.

Amikor elkezdünk begépelni egy parancsot (mondjuk egy READY vagy a RETURN után), a számítógép 'normál' módban van. A vezérlő funkciók, mint például a színválasztás vagy a kurzor mozgatása úgy zajlanak, ahogy eddigi ismereteink alapján elvárjuk. Amint begépelünk egy dupla idézőjelet annak jelölésére, hogy egy füzér kezdődik, a gép átvált idézőjeles módba. Az egyszerű jeleket, tehát a betűket, a grafikai jeleket továbbra is normál módon kezeli, de a vezérlő funkcióknak *nem* engedelmeskedik: ezek bekerülnek a füzérbe 'speciális' karakterként, fordított mezőben megjelenő betűként, jelként vagy grafikus jelként. Amikor a második dupla idézőjelet is begépeljük (így befejezve a fűzést), vagy ha RETURN-t ütünk be, a gép visszavált normál módba.

Kapcsoljuk be a számítógépet, írjunk be egy dupla idézőjelet, aztán egyenként az összes vezérlőjelet. A kapott vezérlőkaraktereket foglaljuk táblázatba:

FUNKCIÓ	BILLENTYŰ LEÜTÉS	KIJELZÉS
Képernyő törlés	SHIFT és CLR HOME	
Kurzor a bal felső sarokba („haza”)	CLR HOME	
Kurzor fel	↑	
Kurzor le	↓	
Kurzor balra	←	
Kurzor jobbra	→	
Fekete	CTRL és 1 BLK	
Fehér	CTRL és 2 WHT	
Piros	CTRL és 3 RED	
Türkizkék	CTRL és 4 CYN	
Bíbor	CTRL és 5 PUR	
Zöld	CTRL és 6 GRN	
Kék	CTRL és 7 BLU	
Sárga	CTRL és 8 YEL	
Narancssárga	C= és 1 ORNG	
Barna	C= és 2 BRN	
Sárgászöld	C= és 3 YLGRN	
Rózsaszín	C= és 4 PINK	
Kékeszöld	C= és 5 BLGRN	
Kék	C= és 6 LBLU	
Sötétkék	C= és 7 DBLU	
Világoszöld	C= és 8 LGRN	
Inverz be	CTRL és RVS ON	
Inverz ki	CTRL és RVS OFF	
Villogás be	CTRL és Flash on	
Villogás ki	CTRL és Flash off	

Próbáljunk ki néhány vezérlő karaktert működés közben. Először ellenőrizzük, hogy a tv készülék színei rendesen vannak-e beállítva, szükség esetén használjuk a TESZTKARTYA programot. Nézzük meg, hogy fehér lesz-e a háttér, amikor begépeljük:

COLOR Ø, 2 RETURN

A következő lépésben írassuk ki a géppel az „EDINBURGH" helynevet kékkel. Írjuk be a következő parancsot:

PRINT" CTRL és 7 BLU EDINBURGH"

(CTRL: tartsuk lenyomva; 7 BLU: míg ezt lenyomjuk)

Most csak annyi jelenik meg a képernyőn (még mindig feketével), hogy

PRINT" ← EDINBURGH". A fordított mezejű ← a

(fordított mezőben)

„kék" jelkódja.

Most üsse le a RETURN billentyűt. Az EDINBURGH név kékkel jelenik meg a képernyőn.

Ebből a példából világosan látjuk az alapelvet: amikor a vezérlő funkciót beírjuk a füzérbe, a gép nem hajtja végre azonnal, csak akkor, amikor a füzér kiírására sor kerül.

Láthatjuk, hogy a villogó kurzor kék színű maradt. Váltsuk vissza feketére úgy, hogy begépeljük a helyes vezérlő utasítást dupla idézőjel nélkül.

A színváltozást kiváltó PRINT parancsot ugyanúgy beépíthetjük a programba, mint bármelyik más utasítást. Írja be és futtassa le a következő programot:

10 PRINT" CTRL és 6 GRN GLASGOW"

20 PRINT" CTRL és 3 RED LIVERPOOL"

30 PRINT" CTRL és 5 PUR ST. CTRL és 8 YEL ANDREWS"

40 STOP

A 30 sor azt bizonyítja, hogy egy füzér több vezérlő funkciót is tartalmazhat; az is kiderül, hogy a fehér háttéren nehéz elolvasni a sárga betűket.

A képernyő- és kurzor-vezérlő funkciókat is beírhatjuk a füzérekbe. Például:

PRINT" SHIFT és CLR HOME ↓ ↓ ↓ → → → CTRL és 3 RED PARIS"

Ez így jelenik meg a képernyőn:

PRINT " ♥QQQ]]] £ PARIS"

(reversed symbols)

Amikor leütjük a RETURN -t, a gép ténylegesen végrehajtja a vezérlő funkciókat. A képernyő le van törölve, a kurzor három hellyel lefelé, és hárommal jobbra tovább lép, és a PARIS szó piros színben megjelenik a képernyő közepe fölötti részben. Próbálja ki magának.

Általában tehát a számítógéppel a képernyőn bárhova írhatunk színes szavakat vagy jeleket, úgy, hogy a megfelelő számú kurzor mozgatást a füzéren belül helyezzük el.

Ha a gép kirajzolt egy képet a képernyőre, nem szeretnénk elrontani a READY-vel és a villogó kurzorral. Ezt úgy oldhatjuk meg, hogy egy 'ciklus stop'-ot, [Más néven *dinamikus* stop vagy állj, vagy halt (ugyanis csak látszólagosan nem fut a program, a valóságban állandóan azt a sort hajtja végre). *(A lektor megjegyzése.)*] azaz egy GOTO utasítást alkalmazunk, amely saját magára ugrik. Amikor a számítógép elérkezik ehhez az utasításhoz, úgy tesz, mint amikor a kutya kergeti a saját farkát, és nem írja ki a READY-t, amíg le nem nyomjuk a RUN STOP billentyűt. Ez a program például fehérrel fogja kiírni a LONDON szót a fekete képernyő közepére:

10 COLOR Ø, 1

20 PRINT" SHIFT és CLR HOME ↓ … ↓ (12-szer) → … → (17-szer) CTRL és 2 WHT LONDON"

30 GOTO 30

Írja be ezt a programot, futtassa le, majd állítsa le a RUN STOP billentyűvel. A képernyő most is fekete lesz, a kurzor pedig fehér, de gyorsan visszajuthatunk az eredeti üzemmódba, ha lenyomva tartjuk a RUN STOP billentyűt, és közben benyomjuk a RESET gombot a számítógép oldalán. Amikor megjelenik a MONITOR szó, írjon be egy X-et, és RETURN Mindig így járjunk el, ha a gép valamilyen okból elakad; jobb mintha ki- és bekapcsolnánk a gépet, mert így a programunk nem vész el.

Rövid gyakorlásként írassunk ki a számítógéppel a képernyő különböző pozícióiba különböző színű szavakat és mintákat. Ne felejtsük, hogy a SHIFT és CLR HOME kitörli a képernyőt, tehát a programban PRINT utasítások sora van, csak az elsőt kell ezzel az utasítással kezdeni – bár a többi közül néhány kezdődhet CLR HOME-al önmagában.

Vége a 9.1 Gyakorlatnak

9.2. Gyakorlat

Tudjuk, hogy a modern órákat kvarckristályok vezérlik, és hosszú időn keresztül rendkívül pontosan járnak. A számítógépben is van kvarckristály, amely több milliószor rezeg másodpercenként, amelynek – többek között – az a feladata, hogy a gép belső digitális óráját vezérelje. Ennek az órának nincs saját számlapja; úgy van kezelve, mint egy füzér változó, azaz kiírathatjuk az időt a képernyőre, amikor akarjuk. Az óraváltozó neve TI$('TIme' = idő).

Amikor először indítunk el egy órát, be kell rajta állítani a pontos időt. A Commodore 16 és Plus/4 gépek sem kivételek. Az órát a billentyűzet segítségével lehet beállítani, a következőhöz hasonló parancs segítségével:

TI$ = "193746" RETURN

amely este 7 óra 37 perc 46 másodpercre állítja be az órát.

Ha nagyon pontosan akarjuk az órát beállítani, akkor várjuk meg a – mondjuk – 8 órás híreket a rádióban. Mielőtt elkezdik a pontos időjelzést, írjuk be:

TI$ "Ø8ØØØØ"

és akkor nyomjuk le a RETURN-t, amikor az utolsó „bip" elhangzik.

Beállítás után a gép belső órája napi néhány másodperces eltéréssel mérni fogja az időt, míg ki nem kapcsoljuk vagy alapállapotba nem hozzuk (reset) a gépet. Az időt nem kell egy programon belül újraállítani vagy megváltoztatni.

Ha kíváncsiak vagyunk a pontos időre, csak a TI$ változót kell elhelyezni egy PRINT utasításban.

Állítsuk be a számítógép óráját a saját óránk szerint (nem lényeges a nagy pontosság). Aztán a PRINT utasítás segítségével jelezzük ki többször a TI$ értékét. Figyeljük meg, hogy hogyan változnak meg a másodpercek a kijelzések közötti időben.

Ezután írassuk ki a pontos időt folyamatosan az alábbi programmal:

```
1Ø PRINT TI$

2Ø GOTO 1Ø
```

Állítsuk le a programot, várjunk néhány percet, majd indítsuk újra. Azt látjuk, hogy az időt most is pontosan mutatja, tehát az óra egész idő alatt működött.

A fenti program időkijelzése nem túlságosan tetszetős. A következő programmal szép digitális órává alakíthatjuk a számítógépet:

a 10-es parancs bíbor keretet választ
a 20-as parancs sárga hátteret választ
a 30-as parancs letörli a képernyőt
a 40-es parancs elmozdítja a kurzort a bal felső sarokba, aztán 9 sorral lejjebb, ott 6 szóközzel arrébb; nincs szükség új sorra.
az 50-es parancs kijelzi kékkel a TI$ értékét
a 60-as parancs visszaugrik a 40-es parancsra

Írja le ezt a programot a lenti keretbe, majd írja be a gépbe, és próbálja ki. Ha valahol *valóban* elakad, keresse meg a helyes megoldást a B. Függelékben, de addig ne futtassa le, amíg nem nézte át alaposan, és rá nem jött arra, hogy működik.

Vége a 9.2 Gyakorlatnak

9.3 Gyakorlat

Alakzat rajzolását gyakran vezérelhetjük ciklussal. Tegyük fel, hogy a bal felső sarokba egy 10 × 10-es piros pontokból álló négyzetet akarunk rajzolni. A négyzethez tíz, egyenként tíz ● grafikus jelet tartalmazó sort kell a képernyőre elhelyeznünk:

```
1Ø PRINT " [SHIFT] és [CLR HOME] ";
2Ø J=1
3Ø PRINT " [CTRL] és [3 RED] ●●●●
●●●●●● "
4Ø J=J+1
5Ø IF J<11 THEN 3Ø
6Ø GOTO 6Ø
```

Ez a program több olyan alapelvet is felhasznál, amelyekkel már találkoztunk a korábbi leckékben. A 1Ø-es parancs végén álló pontosvessző nem engedi, hogy a gép új sort kezdjen a képernyő letörlése után, tehát az első piros pontos sor legfelül jelenik meg. A 2Ø-as és 5Ø-es közötti utasítások alkotják a ciklust, a 6Ø-as pedig egy dinamikus stop.

Gépeljük be a programot, és futtassuk le úgy, ahogy fent leírtuk. Aztán állítsuk le, és próbáljuk ki magunknak, milyen hatása lesz, ha

a) nem tesszük ki a pontosvesszőt a [CLR HOME] után,

b) az 5Ø-es utasításban a 11 helyett más értéket írunk (pl. 15-öt).

c) nem írjuk be a 6Ø-as utasítást.

Ezeket a változtatásokat természetesen listázással és szerkesztéssel lehet végrehajtani. Ne felejtse indulás előtt a kurzor színét visszaállítani kékre vagy feketére!

Ha összefüggő színblokkokat akarunk elérni, inverz szóközöket használunk. Változtassuk meg a 3Ø-as sort:

```
PRINT " [CTRL] és [3 RED] [CTRL] és
[RVS ON] ← 10 szóköz →"
```

és futtassuk le újra a programot.

Mi történik akkor, ha több színtömböt szeretnénk egy képen belül? Az a trükkje, hogy a gép kurzorját a felület első sorához mozgatjuk, aztán kitöltjük a felületet anélkül, hogy érintenénk a képernyőn még meglevő színt. Nézzünk meg két példát:

a) Egy 10 × 10 soros kék blokk rajzolása közvetlenül a piros alatt:
A képernyő alsó fele üres, tehát nem ronthatunk el semmit. Azon kívül a piros tömb megrajzolása után a kurzor a megfelelő helyen lesz. Kibővíthetjük a programot, ha hozzáírjuk:

```
 6Ø J=1
 7Ø PRINT " [CTRL] és [7 BLU] [CTRL]
és [RVS ON] ← 10 szóköz →"
 8Ø J=J+1
 9Ø IF J<11 THEN 7Ø
1ØØ GOTO 1ØØ
```

Figyeljük meg, hogy a dinamikus stop a program végére került. A J-t használtuk vezérlő változóként mind a kék, mind a piros ciklusban, és nincs is vele semmi gond, mert a piros blokkot teljesen befejeztük, mielőtt a kéket elkezdtük volna megrajzolni, így a J-nek *nem* kell egyszerre két feladatot végrehajtania.

b) Egy 10 × 10 soros fekete blokk rajzolása a piros *mellé*. A kezdő vonal a legfelső, ezért a fekete felület megrajzolása közben ügyelnünk kell arra, hogy ne rontsuk el a már megrajzolt piros tömböt. A kurzort hazaléptetjük a bal felső sarokba, és 10 olyan sort csinálunk, amelyeket mindig 10 „kurzor jobbra" mozgatással kezdünk, hogy átugorjuk a piros felületet. Így bővítjük a programot:

```
1ØØ PRINT " [CLR HOME] ";
11Ø J=1
                tízszer
12Ø PRINT " [→] ......... [→]
[CTRL] és [1 BLK] [CTRL] és
[RVS ON] ← 10 szóköz →"
13Ø J=J+1
14Ø IF J<11 THEN 12Ø
15Ø GOTO 15Ø
```

Állítsa össze ezt a programot is, gépelje be és próbálja ki. Számoljon azzal, hogy három különálló ciklus van, amelyeket egymás után hajt végre a gép.

Próbálja meg kiegészíteni úgy a programot, hogy egy bíbor tömböt rajzoljon a fekete alá...

Utolsó gyakorlatként próbáljon meg olyan programokat írni, amelyek az egész képernyőt megtöltik zászlókkal vagy más mintákkal. Szedje össze az eszét, mert különböző csapdák leselkednek ránk:

– A PRINT utasítás végén álló pontosvessző szokványos jelentése az, hogy „Ne kezdj új sort!". Ha a számítógépnek az a feladata, hogy a sor jobb oldalának legszélére tegyen egy karaktert, *automatikusan* elmozgatja a kurzort egy új sor elejére. Ezért amikor a teljes sort ki akarjuk tölteni, pontosvesszőt kell tennünk a végére, hacsak nem akarunk *szándékosan* egy sort üresen hagyni.

– A PRINT utasítással sehogy sem lehet a képernyő jobb alsó sarkába egy karaktert beírni anélkül, hogy az egész képernyő feljebb ne mozogna. Ebben a kockában úgy érhetjük el a kívánt színt, hogy az egész háttérszínt ennek megfelelően választjuk ki.

Gondosan tervezzük meg a képet előre egy kockás papíron. Készüljünk fel arra, hogy a program megírása közben számos hibát vétünk, és ne idegeskedjünk, ha csak többszöri próbálkozásra tudjuk helyrehozni. Jegyezze meg, hogy a sikerélményekből tanul az ember, és nem a kudarcokból, úgyhogy *ne adja fel!*

Indulásnak megadjuk a francia zászló megrajzolását vezérlő programot.

kék	fehér	piros

A középső, fehér sáv 14 karakter szélességű lesz, a két szélső pedig 13 karakternyi. 13+14+13 = 40.

A megfelelő kezdőszín a piros háttér és a fekete keretszín lesz.

25 sorból építhetjük meg a zászlót úgy, hogy mindegyikben 13 fehér és 14 kék négyzet van. Vigyázzunk, hogy az utolsónak eltérőnek kell lennie, nehogy új sor kövesse. Az első 24 sort el tudjuk helyezni egy ciklusban, de az utolsó sorhoz egy külön utasítást kell beírnunk. Tehát erre jutunk:

```
1Ø COLOR 4,1
2Ø COLOR Ø,3
3Ø PRINT " [SHIFT] és [CLR HOME] ";
4Ø J=1
5Ø PRINT " [CTRL] és [RVS ON] [CTRL] és
   [7 BLU] ← 13 szóköz → [CTRL] és
   [2 WHT] ← 14 szóköz →"
6Ø J=J+1
7Ø IF J<25 THEN 5Ø
8Ø PRINT " [CTRL] és [RVS ON] [CTRL] és
   [7 BLU] ← 13 szóköz → [CTRL] és
   [2 WHT] ← 14 szóköz →"
9Ø GOTO 9Ø
```

Futtassa le a programot, és nézze meg alaposan, amíg meg nem értett minden jelet. Most próbáljon meg más zászlókat is megrajzolni, de egyelőre ne kísérletezzen olyanokkal, amelyekben átlós vonalak vannak. Próbálkozzon meg a izlandi zászlóval. A megoldást ellenőrizze a B. Függelékben.

Vége a 9.3 Gyakorlatnak

A 9. Leckéhez tartozó öntesztelő program neve LECKE9KERDESEK.

10. LECKE

10.1 GYAKORLAT

10.2 GYAKORLAT

Az előző fejezetekben rájöttünk arra, hogy egyszer leírt utasításokat a gép többször végrehajthat. Ez történik, ha az utasítást a parancs ciklusban helyeztük el.

Tágabb értelemben ugyanez történik a teljes programokkal is. A programok egy részét használatra tervezik, vagyis kazettákon, lemezeken vagy ún. ROM-modulokban tárolják, hogy bárki többször használhassa. A könyvünkhöz tartozó programokat ilyen példának tekintheti.

Tekintsük először azokat a programokat, amelyeket Ön írt eddig a tanfolyam során. Mindegyiknek ugyanaz a hátránya: akárhányszor futtatjuk le őket, mindig *ugyanazt* az eredményt adják. Ezért aztán nem mondhatók túl hasznosnak!

Hogy példán is bizonyítsuk, menjünk vissza arra a programra, amelyik az angol font (£) és az olasz líra átváltási táblázatát készítette el.

```
1Ø PRINT "£", "LIRA"
2Ø PRINT
3Ø FS=5
4Ø PRINT FS, 235Ø*FS
5Ø FS=FS+5
6Ø IF FS<8Ø THEN 4Ø
7Ø STOP
```

Azon a napon, amikor ez a lecke íródott, a líra fontra való átszámítási kulcsa valóban 2350 volt, így akkor a program helyes eredményt adott. Mára viszont az átszámítási kulcs 2175-re esett vissza. Ha egy bank ma is az eredeti programot használná a líra fontra való átszámításakor, jócskán kiürülnének a tartalékai.

Hogy lehetne ezen a helyzeten változtatni? Egy programozó számára az első kézenfekvő megközelítés az lenne, hogy a 4Ø sort átalakítaná így:

```
40 PRINT FS, 2175*FS
```

Sajnos ezzel az ötlettel nem jutunk messzire! Azoknak a többsége, aki számítógépet használ, nem programozó, de még ha az is, egyszerűen nem érdekli, hogy mi a belseje más valaki programjának.

Ahhoz, hogy a programok rugalmasabbak legyenek, és könnyebben lehessen igazítani őket a mindennapi szükségletekhez, egy további lehetőségre van szükség: arra, hogy a *használó* olyan információt tudjon szolgáltatni, amit a programozó nem tudhatott, amikor megírta a programot. Az olyan programot, amely ezt is lehetővé teszi, rengeteg ember használhatja, és mindenki meg tudja vele oldani a maga sajátos problémáját. Ha például a valuta átszámító program lehetővé teszi, hogy a használó minden alkalmazásakor közölje a pillanatnyi átszámítási arányt, a világ összes bankja tudja használni, és minden elképzelhető átváltási arányra helyes eredményeket hoz ki.

Tegyük fel, hogy valaki másnak tervezünk egy programot. Az elején el kell döntenünk, hogy milyen mennyiségeket fogunk meghatározatlanul hagyni, amelyet a program majd a *használótól* kér megadni.
A példánkban az átszámolási arány egy ilyen mennyiség: a programozó nem ismeri, de ismeri a használó! Az ismeretlen mennyiségeket változókhoz rendeljük, és megfelelő *nevet* adunk nekik. Például az átváltási arányra jó név lehet az AT. Megírhatjuk a programot úgy, hogy szimbolikus neveket használunk a tényleges értékek helyett (amelyeket nem ismerhetünk előre). Így a valuta átváltási program 40 sora:

```
40 PRINT FS,AT*FS
```

Ebből a leírásból természetesen hiányzik még valami. Lehet, hogy *mi* nem ismerjük a változók értékét, de a *gépnek* tudnia kell, amikor a programot futtatja. Az INPUT (adatbevitel) az az utasítás, amellyel a *használó* beviheti a hiányzó információt. A kulcsszót követi(k) a szükséges változó neve(k). Amikor a gép az INPUT utasítást végrehajtja, arra vár, hogy a *használó* beírjon egy értéket, amelyet aztán a gép a névvel ellátott változóban tárol. Így már végre tudja hajtani a program további részét, amely ezt a változót használja.

Mielőtt áttérnénk egy példára, hangsúlyozni szeretnénk egy lényeges kérdést: minden INPUT utasítást tartalmazó programnak pontosan közölnie kell, hogy mit vár a használótól. Ez általában a PRINT utasítással megoldható.

10.1 Gyakorlat

Tanulmányozza alaposan a következő programot:

```
 3 PRINT "KEREM A MAI"
 4 PRINT "ATVALTASI ARANYT"
 5 PRINT "A FONT ES A LIRA KÖZÖTT"
 6 INPUT AT
10 PRINT "£", "LIRA"
20 PRINT
30 FS=5
40 PRINT FS,AT*FS
50 FS=FS+5
60 IF FS<80 THEN 40
70 STOP
```

Vegye észre, hogy a program nem jelöl meg egyetlen konkrét átváltási rátát sem, ezt az értéket a szükséges helyeken az AT képviseli. A program azzal indul, hogy megmondja a használónak, mire van szükség, és megkéri, hogy adja meg az értéket.

Írja be a programot, ellenőrizze gondosan és gépelje be a RUN parancsot. Most *tegyen úgy, mintha* Ön lenne a használó: egy olyan pénzváltó, akinek fogalma sincs a programozásról. A képernyőn megjelenő üzenet arra kéri Önt, hogy írjon be valamit. Amint beírja a megfelelő számot és lenyomja a RETURN billentyűt, a képernyőn megjelenik az átváltási táblázat, amellyel megkezdheti a mai munkáját.

Futtassa le többször is a programot, és figyelje meg, milyen jól kezeli a különböző átváltási arányokat. A program még akkor is értelmes válaszokat adna, ha a lírát újra felértékelnék 23,7-re a fonthoz képest.

Most pedig legyen újra saját maga: azaz a programozó. Amikor a program futtatása alatt megjelent a kurzor, és a gép arra várt, hogy a használó beírja az információját, a gép valójában az INPUT utasításnak engedelmeskedett.

Az INPUT utasítás néhány egymástól kissé eltérő formában használható. Bemutatunk mintapéldákat, és megemlítünk néhány általános szabályt:

1. Törölje a tárat a NEW beírásával, és írja be:

```
1Ø PRINT "HOGY HIVNAK"

2Ø INPUT N$

3Ø PRINT "HELLO [SPACE] "; N$"!"
```

Futtassa le ezt a programot, és nézze meg, mi történik. A példából látható, hogyan működik az INPUT utasítás füzérekkel és hasonlóan számokkal. [Számként – és nem számjegyekből álló füzérként – akkor vár választ, ha a változó numerikus az INPUT utasításban *(a lektor megjegyzése).*] Amikor azt akarjuk, hogy a számítógép 'barátságos' legyen a használóval, használjuk ezt vagy hasonló sorozatot a program elején. Ha a program valamilyen kitalálós játék (kvíz), az N$ értékét használhatjuk pl. ilyen utasításokban:

```
4Ø PRINT "NO [SPACE] "; N$;"MEGY EZ
   NEKED JOBBAN IS"
```

(Ha nem tudja eldönteni, hogy mit ír a képernyőre ez az utasítás, írja hozzá a már gépben levő programhoz és futtassa le újra.)

2. Próbálja meg ezt:

```
1Ø INPUT "NEV";N$

2Ø PRINT "VISZLAT [SPACE] "; N$
```

Ebben a példában azt látjuk, hogy hogyan lehet az INPUT utasításban elhelyezni egy rövid eligazító információt. Az információ a használó segítségére jelenik meg a képernyőn, közvetlenül a ? előtt.

A példában szereplő 1Ø-es utasítás pontos megfelelője ennek a sorozatnak:

```
PRINT "NEV";

INPUT N$
```

Ne feledje, hogy a leíró szavakból álló füzér után pontosvesszőnek *kell* lennie.

3. Végül próbálja ki az alábbi programot:

```
1Ø PRINT "KEREK KET ÖSSZEADANDO
   SZAMOT"

2Ø INPUT A,B

3Ø PRINT "A SZAMOK
   OSSZEGE=";A+B

4Ø STOP
```

Az INPUT utasítás két értékre vár, a használónak úgy kell beírnia, hogy vagy vesszőt ír közéjük, vagy lenyomja a [RETURN] billentyűt. Tehát a begépelendő adat például 43,19

vagy $\begin{cases} 43 \\ 19 \end{cases}$

Tulajdonképpen az INPUT utasítás akárhány változót kérhet a használótól, de a bonyodalmak elkerülése végett jobb, ha kettőnél maradunk. Az utasításon belül a változókat vesszők választják el egymástól.

Ha néhányszor már lefuttatta a programot, írjon be képtelenségeket, mint pl. DONALD, KACSA

A számítógép bármit elfogad füzérnek, de ha számra van szüksége, és olyasmit kap, ami nem igen lehet szám, ezt az üzenetet írja ki:

REDO FROM START
(újra kérem az adatokat)

és újabb lehetőséget ad.

Adódhat olyan helyzet is, hogy le akarjuk állítani a programot, miközben egy INPUT utasítást hajt végre és kijelzi a kurzort.

Ilyenkor a [RUN STOP] billentyű hatástalan, a leállást csak sokkal bonyolultabban lehet megoldani. Tartsa lenyomva a [RUN STOP] billentyűt, és nyomja meg a RESET gombot. Ekkor azt látja majd, hogy a kurzor megjelenik az üres képernyőn, amelyen csak a MONITOR szó, néhány betű és szám áll.

Most gépelje be: X [RETURN] és a READY üzenet megjelenik, ami azt jelzi, hogy a gép újra vezérelhető.

Arra nagyon vigyázzon, hogy a RESET gombot önmagában ne nyomja le, mert azzal tönkretenné az egész programot.

Vége a 10.1 Gyakorlatnak

10.2 Gyakorlat

Könnyű hasznos programokat írni, ha nem tévesztjük szem elől, hogy a programozó és a használó két *különböző* ember. Nem tételezhetjük fel, hogy a használó ért a programozáshoz (ezért nem várhatjuk el tőle, hogy kilistáztassa a programot, ha rá akar jönni, miről is van szó). A programozó nem tud „beszélgetni" a használóval, csak a számítógéppel küldhet üzeneteket a képerőnyőre; a használó különösen nem mehet vissza egyáltalán a programozóhoz, így hát jobb, ha semmi nem marad a programban megválaszolatlanul.

Amikor programot tervez, képzelje magát egy olyan kívülállónak, aki figyel valakit, aki próbálja azt használni.

Gondolja át az összes hibalehetőséget, és próbálja kizárni azokat azzal, hogy a használónak minden lehetséges útmutatást megad.

Amikor megírta a programot, vegye át a használó szerepét, később pedig, utolsó tesztként, vegye rá egy barátját vagy rokonát, hogy „kísérleti nyúlként" próbálja ki a programot. Ha a kísérleti nyúlnak meg kell kérdeznie, hogy mit kell tennie, vagy hogy mit is jelentenek valójában a kijelzett válaszok, akkor a teszt nem sikerült, és ennek megfelelően újra kell tervezni a programot.

Írjon olyan programot, amely az alábbi feladatot hajtja végre:

a) Elkészíti a használó által kiválasztott szám szorzótábláját.

b) Megkérdezi a használótól (aki feltételezésünk szerint házasember) a családi nevét, aztán a felesége keresztnevét, aztán írassuk ki a programmal a felesége teljes nevét. [Feltéve, hogy névlehetőségként a feleség a férj családi nevét és saját keresztnevét válaszotta *(a lektor megjegyzése).*]

A megoldásokat a B. Függelékben találja, de addig ne nézze meg azokat, míg meg nem tett mindent azért, hogy önállóan írja meg a programokat.

Vége a 10.2 Gyakorlatnak

A 10. leckéhez tartozó kitalálós játék (kvíz) neve LECKE10KERDESEK.

11. LECKE

11.1 GYAKORLAT

11.2 GYAKORLAT

11.3 GYAKORLAT

A programozás egyik legérdekesebb tulajdonsága gazdagsága és változatossága. Ugyanaz a számítógép megfelelően programozva, működhet kalkulátorként, oktatógépként, hangszerként, kórházi beteget ellenőrző monitorként vagy szinte bármi másként, ami csak eszünkbe jut. Ez a széles körű alkalmazhatóság abból adódik, hogy néhány alaputasítást többféleképpen kombinálhatunk.

Eddigi programjainkban mindössze 7 utasítást használtunk:

PRINT, LET, GOTO, IF, INPUT, STOP és

COLOR.

Természetesen van még néhány BASIC utasítás, amelyekkel ezután kell megismerkedni. Ebben a leckében viszont azoknak az utasításoknak a felhasználhatóságát mérjük fel, amelyekkel már megismerkedtünk.

Az összes közül a legrugalmasabb utasítás az IF. Az előző leckékben különböző ciklusok vezérlésére használtuk, de számos egyéb módon is alkalmazható. Például meg tudja vizsgálni a használótól kapott *adatok* vagy tételek érvényességét, és ezzel helyes irányba vezérli a számítógép tevékenységét.

11.1 Gyakorlat

Képzelje el, hogy Ön egy számítógépes házasságközvetítő irodát nyit, és első szolgáltatásként egy olyan programot akar tervezni, amely az ügyfelek kor szerinti választásában nyújt segítséget. A hagyomány azt tartja, hogy a férfinak olyan nőt kell feleségül vennie, akinek az életkora a férfiénak a fele plusz hét év. Ebből logikusan következik, ha utána gondol, hogy a nőnek olyan férj való, aki nála kétszer idősebb, mínusz 14 év.

A tanácsadó programnak nyilvánvalóan abból kell kiindulnia, hogy megtudakoljuk a házasulandó életkorát. Ahhoz, hogy a program használható legyen, azt is meg kell tudnunk, hogy a házasulandó férfi vagy nő. A programot férfiak is, nők is fogják használni, ezért a két nemre vonatkoztatva két külön utasításcsoportnak kell a programban lennie. És végül egy IF utasításra is szükség lesz, amely a konkrét helyzetnek megfelelően kiválasztja a szükséges csoportot.

Itt adjuk a tanácsadó program első változatát. Olvassa át alaposan, és indokolja meg, hogy miért van szükség az egyes utasításokra.

```
10 INPUT "HANY EVES"; EV

20 INPUT "FERFI VAGY NO"; NM$

30 IF NM$="FERFI" THEN 70

40 PRINT "ON KERES"

50 PRINT "EGY FERFIT AKI"; 2*EV-14

60 STOP

70 PRINT "TALALNIA KELL"

80 PRINT "EGY LANYT AKI"; EV/2+7

90 STOP
```

Nyilván rájött, hogy az EV változó az ügyfél korát, az NM$ pedig azt jelöli, hogy férfi vagy nő az ügyfél. Az NM$=„FERFI" feltétel igaz, ha a házasulandó FERFI-val válaszol a "FERFI VAGY NO" kérdésre. Az EV/2+7 BASIC nyelven azt jelenti, hogy „az évei számának fele plusz hét", míg a 2*EV–14 kifejezés azt jelenti, „kétszer az évei száma mínusz tizennégy".

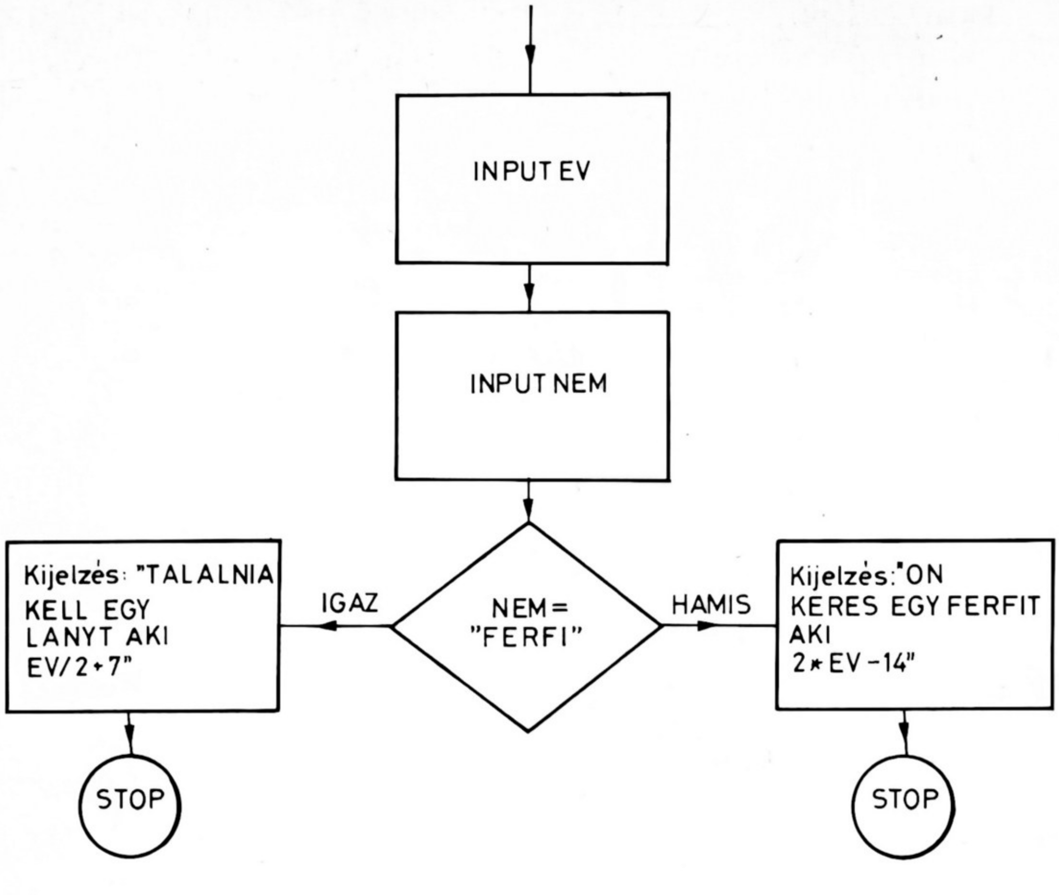
Amikor átnézte a programot, győződjön meg arról, hogy valóban érti-e, azzal, hogy meg tudja-e *előre mondani,* hogy mi fog megjelenni a képernyőn: a) egy 20 éves férfi, és b) egy 22 éves nő esetében. Használja a következő táblázatot (az elsőt részben kitöltöttük).

a)	b)
RUN HANY EVES? 20 FERFI VAGY NO?	RUN

Írja be a programot a számítógépbe. Próbálja ki a különböző ügyfelek adataival, és ellenőrizze, hogy helyes volt-e mindkét előrejelzése.

Ez az egyszerű példa azt bizonyítja, hogy a programozónak nem kell a számítógép működését előre meghatároznia, hanem megoldhatja, hogy a használó információjától függően dolgozzon.

Mivel a programoknak gyakran bonyolult döntéseket kell hozniuk, a megtervezésükhöz egy *folyamatábrának* nevezett ábrát használunk. A házassági tanácsadó program folyamatábrája így néz ki:



Egy folyamatábra különböző alakú „dobozok" vonallal jelölt kapcsolata. A kapcsolatok irányát nyilakkal fejezzük ki. Négyféle „doboz" van:

a) A *négyzet* vagy *téglalap* alakú doboz tartalmazza valamely egyszerű cselekvésnek a leírását, amelyet később egy vagy két BASIC utasításra le lehet fordítani. A mi példánkban a felső két „doboz" ilyen típusú. A nyíl jelöli, hogy a program az első „doboz" végrehajtásával kezdődik, és a nyíl irányában sorban megy tovább a másodikra.

b) A *rombuszba* feltételt helyezünk, amely vagy igaz, vagy hamis. A rombusz alakba egy vonal vezet be, de kettő vezet ki belőle, IGAZ és HAMIS felirattal (ez lehet IGEN és NEM is). A rombusz tehát IF utasításnak felel meg. Arra utasítja a számítógépet, hogy vizsgálja meg a feltételt, és az eredménynek megfelelően az IGAZ vagy a HAMIS ágon haladjon tovább.

c) A *kör* a végblokk, amely arra utasítja a számítógépet, hogy hagyja abba a program végrehajtását. A körben STOP szó áll.

d) A *felhő** (amely nem szerepel a példánkban) azt a cselekvést jelképezi, amely túl bonyolult ahhoz, hogy részletesen leírjuk. A felhőt általában egy külön, teljes folyamatábrán lehet részletezni, mint ahogy az országos térképhez is mellékelik a városok részletes térképét.

A folyamatábra valójában a program „térképe". A programot futtató számítógép majdnem olyan, mint egy táblajátékot játszó ember. A játékos bábuja először rámegy az első ‚kockára'. Amikor a kockában leírt cselekvés lezajlott, a bábu a nyíllal jelölt vonal mentén mindig átmegy a következő kockába.

Amikor a bábu beér a rombuszba, a játékos eldönti, hogy igaz-e a feltétel. Ha igaz, továbbküldi a bábut az IGAZ vonal végén levő dobozba, ill. a HAMIS vonalat követi. Végül elérkezik a STOP-hoz, amely a játék végét jelenti. Ezzel a példával két olyan dologra szeretnénk felhívni a figyelmet, amelyet a számítógépről feltétlenül tudni kell:

* Míg az előző folyamatábra elemek szabványosak, a ‚felhő' nem. Célszerűbb használni kettős vonallal határolt téglalapot: *(a lektor megjegyzése).*

– A számítógép egyszerre csak *egy* dolgot tud végrehajtani (többet nem).

– A program határozza meg, hogy a számítógép milyen sorrendben hajtja végre a tevékenységeket.

Sokakat meglep, hogy egy egyszerű GOTO utasításnak nincs a folyamatábrán külön jelzése. Ez azért van, mert a GOTO egyáltalán nem határoz meg cselekvést, csak az utasítások végrehajtási sorrendjét befolyásolja. Ezt pedig jól tudjuk szemléltetni egy összekötő vonallal. Például:

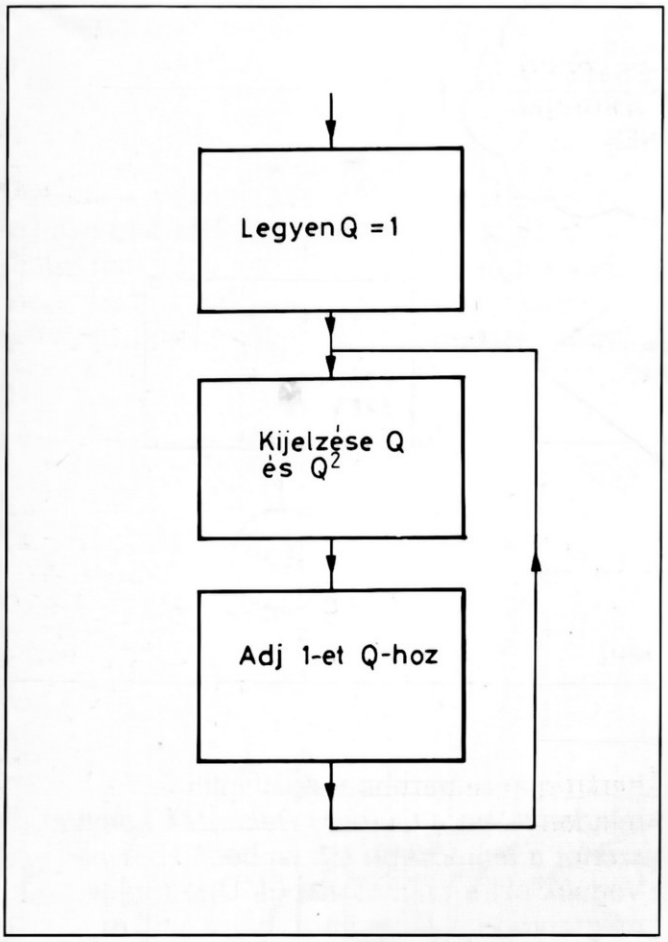
```
1Ø Q=1

2Ø PRINT Q; Q*Q

3Ø Q=Q+1

4Ø GOTO 2Ø
```

folyamatábrája:



És most rajzolja meg az alább következő program folyamatábráját. [Ebben segíthet egy ún. szervezői vonalzó (sablon), amely írószerboltokban kapható *(a lektor megjegyzése).*]

```
1Ø S=1

2Ø PRINT S, 12*S

3Ø S=S+1

4Ø IF S<13 THEN 2Ø

5Ø STOP
```


(Ellenőrizze a megoldást a B. Függelékben!)

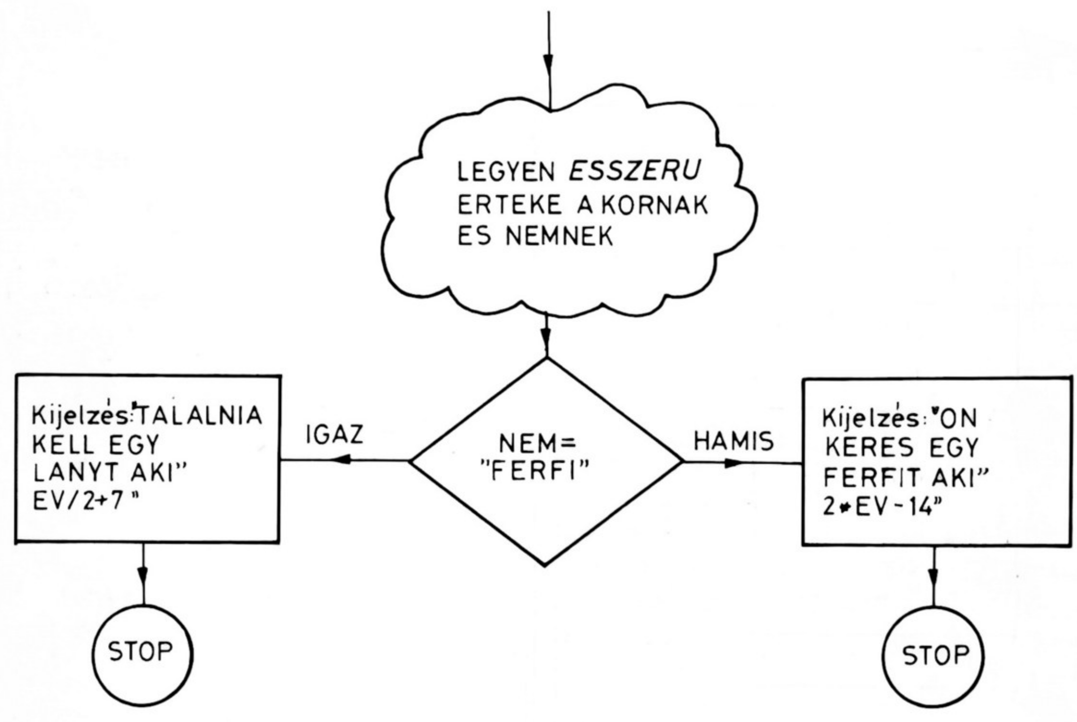
Vége a 11.1 Gyakorlatnak

11.2 Gyakorlat

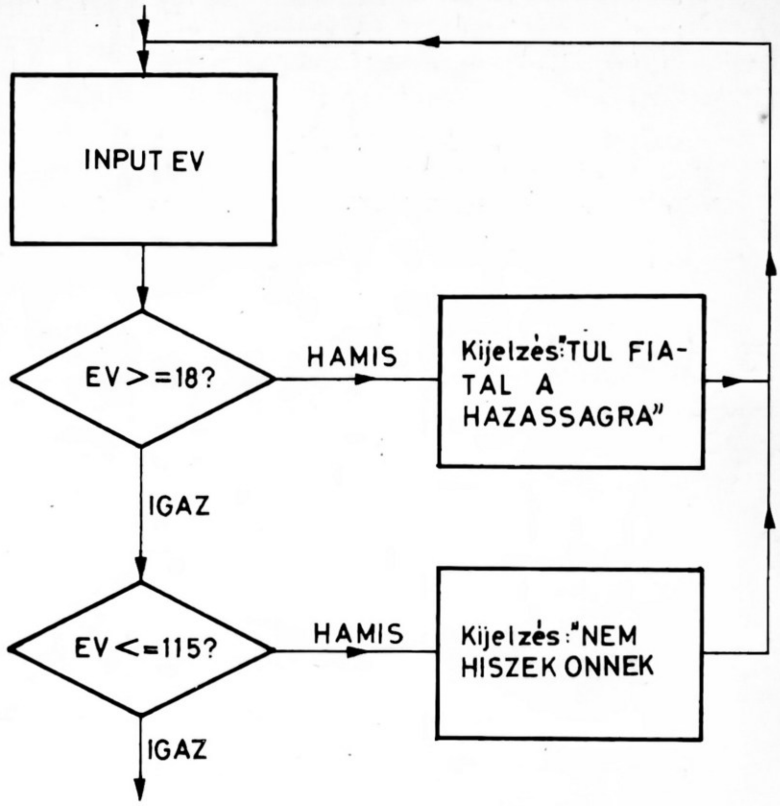
Folytassuk a felderítő munkát. A házassági tanácsadó programunk egyik jellemzője, hogy ha helytelen adatokkal látjuk el, bolondos válaszokat ad. Például egy hat éves kislánynak azt tanácsoljuk, hogy keressen magának egy mínusz két éves férjet, aki még a szülők gondolatában sem fogant meg. További probléma lehet, ha a használó bármilyen szót ad válaszul a második kérdésre, amely nem a FERFI szó, a program őt nőnek tekinti. Annak, aki úgy válaszol, hogy „F", vagy „FIU", a gép azt fogja tanácsolni, hogy találjon egy *férfit* házastársnak.

Sok olyan program van, ami ilyen buta módon viselkedik, és ezek bizony lejáratták a számítástechnikát. A gyakorlatban ezeket a kellemetlenségeket elkerülhetjük azzal, hogy a használó által adott információt átengedjük egy olyan szűrőn, amely biztosítja, hogy a válasz legalábbis *ésszerű* legyen.

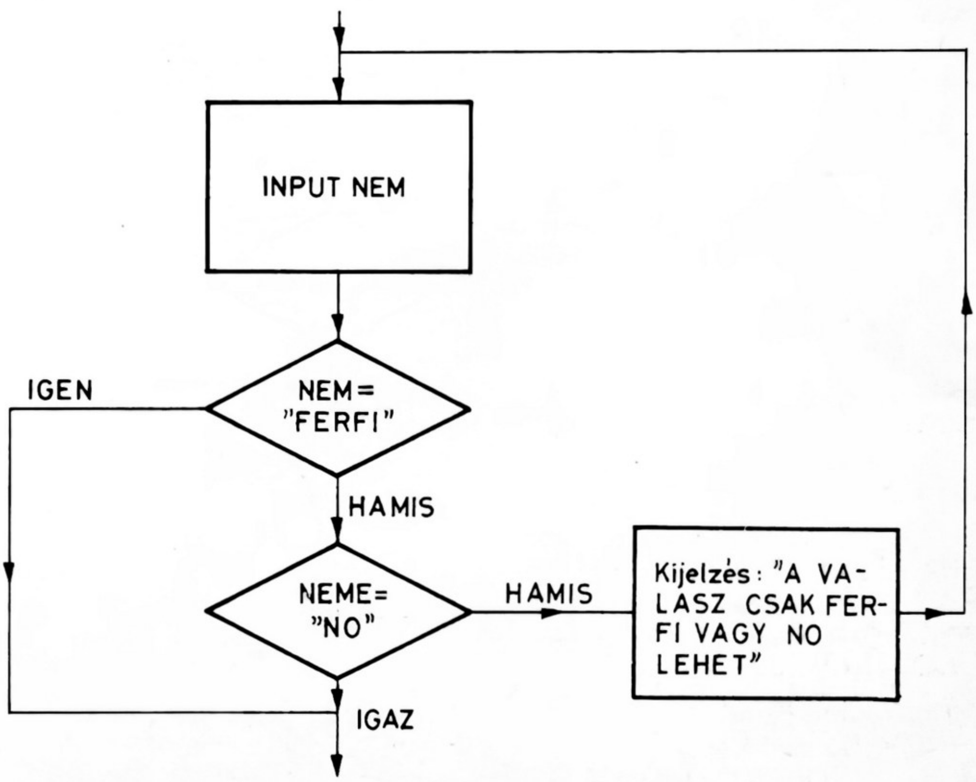
Először is rajzoljunk egy új folyamatábrát az egész programnak, a részletes input dobozokat egy felhővel helyettesítve:



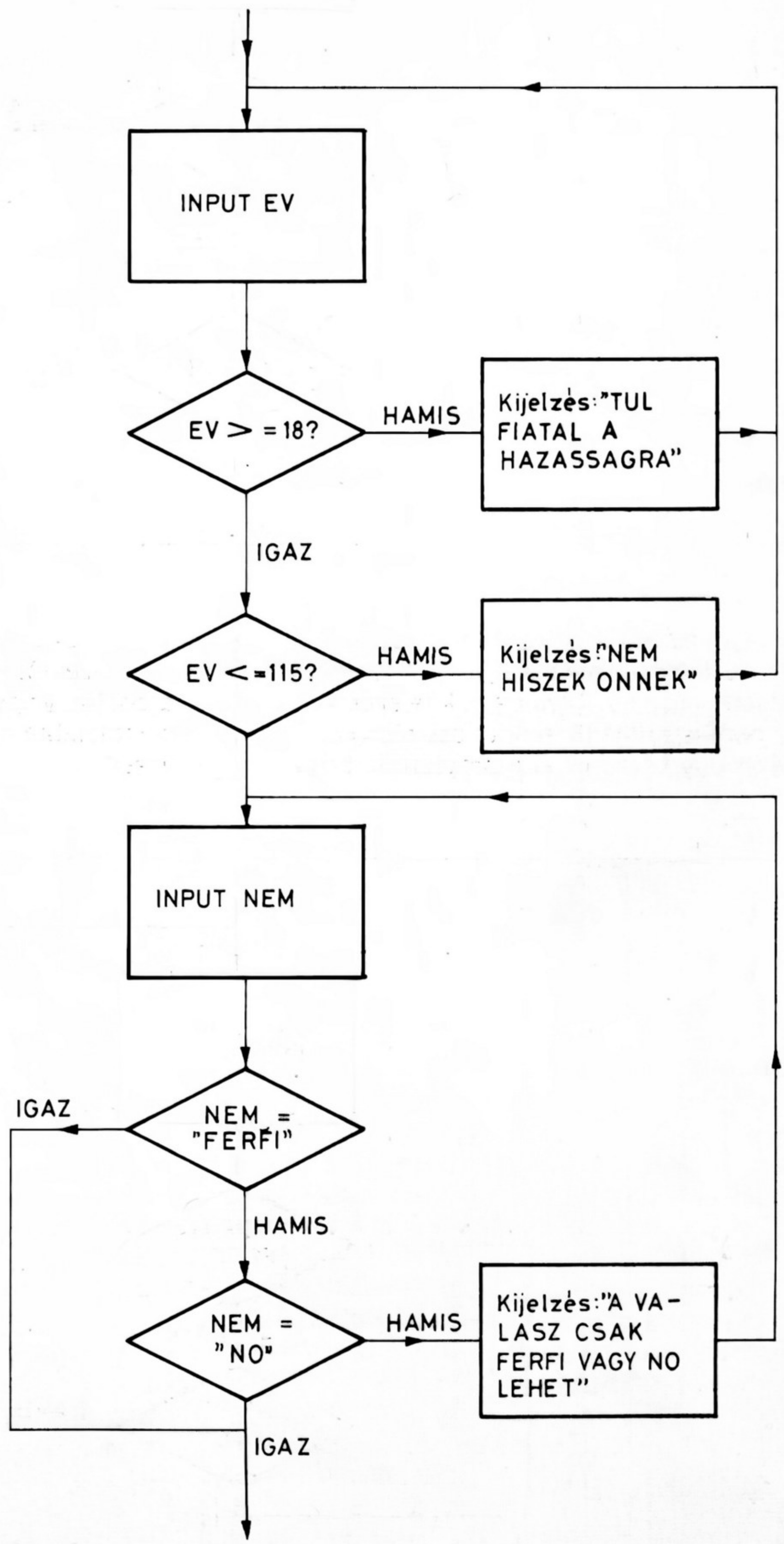
Azért használjuk itt a felhőt, mert még nem határoztuk meg részletesen, hogy mit értünk pontosan „ésszerű" válasz alatt. A felhő azért hasznos, mert így a program egészét meg tudjuk tervezni, viszont azt a kötelezettséget is jelenti, hogy a cselekvéssorozatot részletesebben kidolgozzuk. Ebben a stádiumban a tervezés még nincs befejezve, de ez nem jelenti azt, hogy a fő folyamatábra hasznavehetetlen vagy rossz lenne! Lássuk, mit jelent az, hogy „ésszerű" válasz. Vegyük először a használó életkorát. A legkisebb elfogadható érték 18, mert 18 éven aluli emberek nem túl gyakran házasodnak. A felső határt már nehezebb megállapítani, mindenesetre a *Guiness Rekordok könyve* szerint a legidősebb élő ember 115 éves. Vegyük ezt a számot alapul. Úgy fogjuk megtervezni a programot, hogy amikor a számítógép megtudakolja az ügyfél életkorát, döntse is el, hogy az elfogadható-e ésszerű válaszként. Ha úgy dönt, hogy nem, kiírja az okot, és megkéri az ügyfelet, hogy elfogadhatóbb értéket adjon. Nézzük meg a folyamatábrát:



Amikor a második kérdésre kerül sor, a válaszoló többféleképpen tudja közölni, hogy férfi vagy nő. Olyan sok kifejezés van, hogy nem is tudnánk mindet számbavenni. Ehelyett úgy készítjük el a programot, hogy csak két szót „értsen": FERFI és NO. Ha a válasz ezektől eltérő, a program arra kéri a használót, hogy ismételje meg a választ. A folyamatábra megfelelő része ilyen lesz helyesen:



Most összerakhatjuk a két részt, hogy megkapjuk a felhő teljes folyamatábráját, amely értelmes értékeket ad az EV és az NM$ változóknak.



Egy gondosan előkészített folyamatábra halmaz alapján nagyon könnyű megírni a programot. Először a fő folyamatábrán indulunk el, de mivel az első blokk egy felhő, áttérünk a hivatkozott kiegészítő folyamatábrára és lefordítjuk. Ezután visszatérünk a fő folyamatábrára a program maradék részére.

Figyelje meg, hogy mindkét kiegészítő folyamatábrán a rombusz igaz ága a program végéhez vezet. A megfelelő IF utasítás sorszámait úgy tudjuk a legegyszerűbben kitölteni, ha egyelőre REM-et teszünk a felhő végére, és ennek sorszámát használjuk fel az IF utasításban. A REM semmit nem tesz, csak kényelmes kikötőhelyként szolgál:

```
10 INPUT "HANY EVES"; EV
20 IF< = 18 THEN 50
30 PRINT "TUL FIATAL
   A HAZASSAGRA"

40 GOTO 10

50 IF EV< = 115 THEN 80

60 PRINT "NEM HISZEK ONNEK"

70 GOTO 10

80 INPUT "FERFI VAGY NO"; NM$

90 IF NM$ = "FERFI" THEN 130

100 IF NM$ = "NO" THEN 130

110 PRINT "A VALASZ CSAK FERFI
    VAGY NO LEHET"

120 GOTO 80

130 REM EV ES NM$ ESSZERU
    ERTEKET KAPOTT
```

Ezt követi a programnak az a része, amely már korábban készült (de kiigazított sorszámokkal):

```
140 IF NM$ = "FERFI THEN 180

150 PRINT "ON KERES EGY"

160 PRINT "FERFIT AKI"; 2 * EV - 14

170 STOP

180 PRINT "TALALNIA KELL EGY"

190 PRINT "LANYT AKI"; EV/2 + 7

200 STOP
```

Gépeljük be a programot, és próbáljuk ki. Ésszerű értékekkel úgy fog működni, mint az első változat, de sokkal jobb lesz a buta válaszok felfedezése és kizárása terén. Megvan az a fontos tulajdonsága, hogy megbízható, illetve az a képessége, hogy nem lehet félrevezetni.

A lecke befejezéseül készítsen az Olvasó önállóan egy programot. A munka megkezdése előtt adunk néhány tanácsot:

1. Vegyen elő tiszta papírt, ceruzát és radírt. Kapcsolja *ki* a számítógépet.

2. Tanulmányozza alaposan a feladatot, és dolgozzon ki önállóan egy-két egyszerű példát. A megoldásokat ellenőrizze a számítógéppel.

3. Kezdje azzal, hogy eldönti, milyen változókra van szüksége. Írja le a nevüket, típusukat és céljukat egy „változójegyzékbe". A tanácsadó program változóit például így lehetett volna bejegyezni:

Név	*Típus*	*Cél*
EV	Szám	Az ügyfél kora
NM$	Füzér	az ügyfél neme, azaz "FERFI" vagy "NO"

4. Rajzolja meg a program folyamatábráját. Készüljön fel rá, hogy sokat fog hibázni, és ne lepődjön meg, ha ötször-hatszor újra kell rajzolnia az ábrát. Tartson ki, amíg elégedett nem lesz az eredménnyel. Programozni nehéz, és a munkának ehhez a részéhez – a folyamatábra elkészítéséhez – kell a legtöbb energia.

5. Most fordítsa le a folyamatábrát BASIC nyelvre. Ennek már könnyen kell mennie. Ha mégis nehézségei támadnak, akkor biztosan nem jól készítette el a folyamatábrát, célszerű mindent elölről kezdeni.

6. És most – végül – kapcsolja be a számítógépet, és írja be a programot. Attól a néhány gépelési hibától eltekintve, ami most becsúszhat, gond nélkül le kell futnia a programnak. Gyakorolja az új módszert annyi különböző példán, amennyit csak ki tud találni; térjen vissza a már korábban kidolgozott példákra is. Őrizze meg

a programját szalagon vagy lemezen (ha meg akarja őrizni), tegye el a folyamatábrákat és a változójegyzéket is.

Most azt a munkamódszert ismertettem, amivel egy jó szakember végzi a programozást. Sok ember nem így dolgozik; leül a számítógép elé, és egyenesen a billentyűzeten állítja össze a programot. Ez a módszer néha be is válik, ha apróbb problémákat kell csak megoldani, de általában hosszú, értelmetlen programokká vezet, amelyek csak részben működnek, és amelyeket a programozó nem képes megváltoztatni vagy rendbehozni. Az is sokkal több időbe telik, míg valahogy valamit működésbe hozunk. Ez a tény egyáltalán nem nyilvánvaló – gyorsabbnak tűnik a tervkészítést kihagyva rögtön munkához látni. Valójában ez az oka annak, hogy sokan olyan rosszul programoznak.

Ön is választhat: vagy úgy dolgozik, ahogy tanácsoltuk, és akkor rövid időn belül hozzáértő programozó lesz; vagy a kanyargósabb és lassúbb utat járja.

Most tervezze meg, készítse el a folyamatábráját, írja meg és tesztelje le az alábbi problémát megoldó programot:

Ruritaniában az alábbi elvek szerint vetik ki a házadót:

Minden ajtóra	£ 57
Minden ablakra	£ 12
Minden zsúptetőre	£ 38
Minden cseréptetőre	£ 94

Tételezze fel, hogy a házak csak zsuptetősek vagy cseréptetősek lehetnek, írjon olyan programot, amely minden ház részletes adatait megtudakolja, majd megadja a házanként fizetendő adó összegét. Például egy egyajtós, kétablakos zsuptetős ház esetében a helyes megoldás £ (38 + 57 + 2 * 12) = £ 119.

Számoltassa ki a programmal a lerajzolt házak adóját (feltételezve, hogy minden ajtó és ablak a ház elején van):



Ide rajzoljon:

Ellenőrizze a válaszait a B. Függelékben.

Vége a 11.2 Gyakorlatnak

11.3 Gyakorlat

Töltse be és futtassa le
a LECKE11PROGRAM nevű programot.
Amikor kilistázta, vizsgálja meg az utasításokat, rajzolja meg a folyamatábráját és készítse el a változójegyzéket.

Vége a 11.3 Gyakorlatnak

12. LECKE

12.1 GYAKORLAT

12.2 GYAKORLAT

Általában nem kell sokat keresgélni, hogy egy programban ciklust találjunk.
A programozásban a ciklusok annyira hétköznapiak és annyira fontosak, hogy az alapvető részeinek leírása a BASIC nyelv „gyorsírásos" módszerrel is rendelkezik. Remélhetőleg emlékszik még az Olvasó, hogy minden ciklus vezérléséhez négy alapvető dologra volt szükségünk:

– A ciklusváltozó kiválasztása.
– A ciklusváltozó kezdőértéke.
– A ciklusváltozó utolsó vagy végértéke.
– A növekmény, vagyis az a mennyiség, amennyivel a ciklusváltozó lépésenként növekszik.

Mindezen részeket helyettesíteni lehet egy speciális utasítással, amelynek a kulcsszava a FOR (FOR = -nak, -nek.). Nincs is másra szükség, kivéve egy NEXT (NEXT = következő.) utasítást, amely a ciklusmag végét jelzi.

A következő két programnak pontosan ugyanez az eredménye; hasonlítsuk össze:

```
1Ø J=4
2Ø PRINT J,J*7
3Ø J=J+2
4Ø IF J < 22 THEN 2Ø
```

(Az IF...THEN használatával)

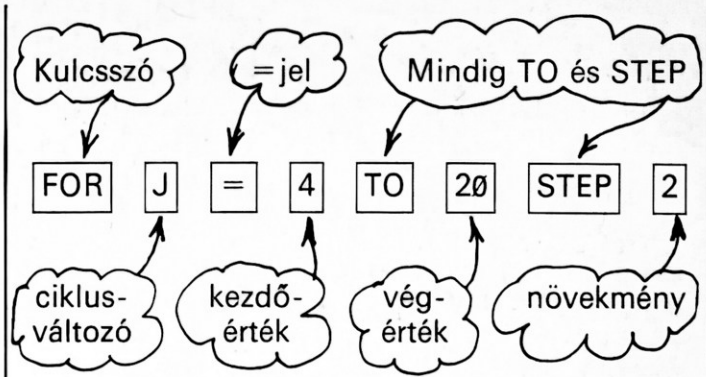
```
1Ø FOR J=4 TO 2Ø STEP 2*
2Ø PRINT J,J*7
3Ø NEXT J
```

(A FOR...NEXT használatával)

Mindkét esetben:

A ciklusváltozó	J
A kezdőérték	4
A végérték	2Ø
A növekmény	2

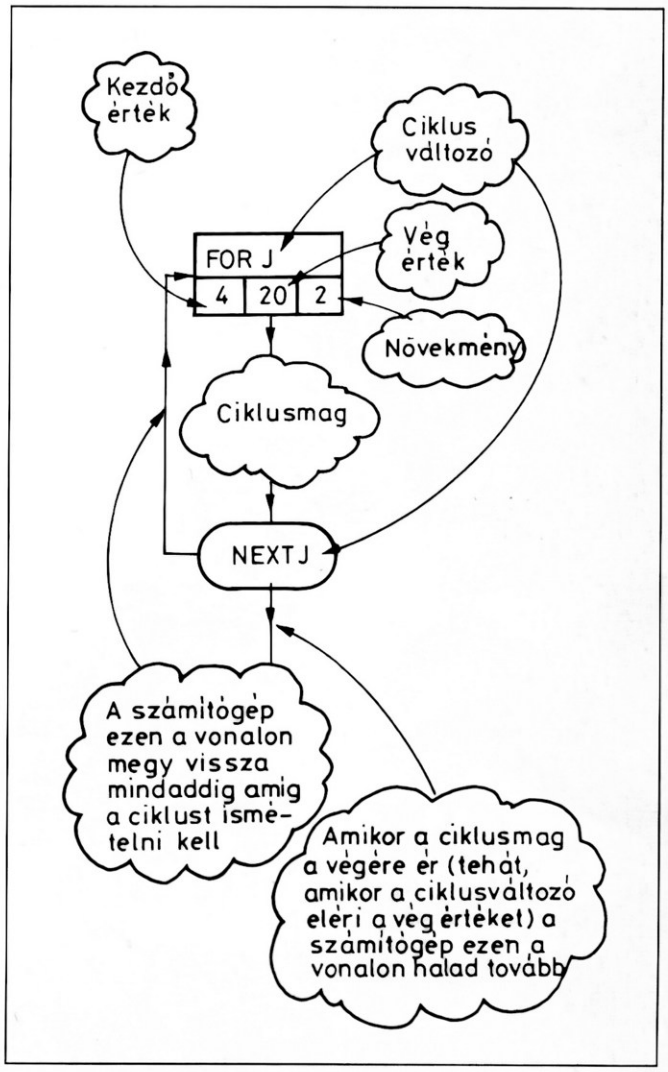
A következő példa azt mutatja be, hogyan épül fel a FOR utasítás:



A NEXT utasítás közli a ciklusváltozó nevét mintegy ellenőrzésként, segítve a program olvasását.

Minden FOR utasításhoz tartoznia kell egy NEXT utasításnak, és a kettő közé van *zárva* a ciklusmag.

A folyamatábrákon a ciklusokat különleges módon jelöljük, olyan jelöléseket alkalmazva, amelyek nehezen téveszthetők össze más tevékenységfajtákkal:



* TO = -ig, STEP = lépés, vagyis a FOR J = 4 TO 2Ø STEP 2 így lenne olvasható: „A J-nek adott 4 értéktől a 2Ø értékig 2-esével." *(A lektor megjegyzése.)*

12.1 Gyakorlat

Segítse elő az emlékezetébe vésni a FOR utasítás részleteit azzal, hogy átnézi ezeket a rövid gyakorlatokat, és leírja, hogy Ön szerint mit ír ki a számítógép. Végül ellenőrizze a válaszait a számítógépen:

(1)
```
10 FOR Q=1 TO 16 STEP 5
20 PRINT Q;
30 NEXT Q
40 STOP
```

Az Ön megítélése szerint:

(2)
```
10 FOR R=38 TO 50 STEP 3
20 PRINT R; 50-R
30 NEXT R
40 STOP
```

Az Ön megítélése szerint:

Fordítsa le a következő programot FOR – NEXT jelölési módra. Úgy ellenőrizze a válaszát, hogy mindkét változatot lefuttatja a számítógépen – a két válasznak azonosnak kell lennie:

```
10 PRINT "KILENCES SZORZOTABLA"
20 S=1
30 PRINT S;"-SZER 9=";
40 PRINT 9*S
50 S = S+1
60 IF S < 13 THEN 30
70 STOP
```

Az Ön megoldása:

A FOR és NEXT utasítással kapcsolatban meg kell jegyezni néhány dolgot:

a) Ha a növekmény vagy a lépések száma 1, a „STEP 1"-et el lehet hagyni a FOR utasítás végéről. A számítógép tudja, hogy ez mit jelent.

b) A ciklusvezérlést úgy is meghatározhatjuk, hogy *hátrafelé* számláljon: ilyenkor negatív lépésszámot használunk. Az alábbi program:

```
10 FOR X = 10 TO 5 STEP - 1
20 PRINT X;
30 NEXT X
```

futtatási eredménye: 10 9 8 7 6 5, ilyen sorrendben.

c) A ciklusmagot legalább egyszer mindig végrehajtja a gép, még akkor is, ha a végérték kisebb, mint a kezdőérték. Például:

```
10 FOR R = 5 TO 3
20 PRINT R
30 NEXT R
```

ezt fogja kijelezni: 5.

d) A FOR utasításban megadott értékeknek nem kell feltétlenül számnak lenniük, lehetnek olyan kifejezések, amelyek más változókat tartalmaznak. A következő program például annyi szívet fog kijelezni, ahányat *a használó kér.*

Próbálja ki, és elemezze alaposan:

```
10 INPUT "MENNYI SZIV"; SZ

20 FOR K=1 TO SZ

30 PRINT" CTRL és RED ♥"

40 NEXT K

50 STOP
```

e) A ciklusváltozó nem lehet füzér. Például ez az „utasítás":

~~FOR X$="A"TO"ABBBB"STEP"B"~~

NEM BASIC NYELVEN VAN

Szintaktikus hibát (SYNTAX ERROR) eredményezne, vagyis ezt a szerkezetet nem szabad használni. Mindezek ismeretében mondja meg előre a következő programok eredményét, és ellenőrizze a számítógépen:

(i)
```
10 FOR A=1 TO 4
20 PRINT A*A;
30 NEXT A
40 STOP
```

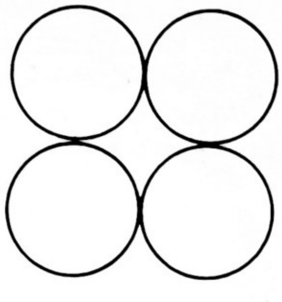
(ii)
```
10 FOR B=3 TO 0 STEP - 1
20 PRINT B;
30 NEXT B
40 STOP
```

(iii)
```
10 FOR C = 5 TO 4
20 PRINT C;
30 NEXT C
40 STOP
```

(iv)
```
10 X = 5
20 Y = 9
30 Z = 2
40 FOR W = X TO Y STEP Z
50 PRINT W;
60 NEXT W
70 STOP
```

Eddig minden figyelmünket a FOR és NEXT utasítások részleteire fordítottuk, ezért a lehető legegyszerűbb ciklusmagokat választottuk. A gyakorlatban a ciklusmagnak nem kell rövidnek és egyszerűnek lennie, olyan összetett is lehet, amilyet csak akarunk – egy dolgot kell csak észben tartani: a számítógép mindannyiszor végrehajtja a benne foglalt utasításokat, ahányszor áthalad a cikluson.

Nézzük a következő feladatot: ágyúgolyókból építsen fel egy négyzetes alapú piramist. Fentről kezdve 1, 2, 3...-mal fogjuk számozni a szinteket. Az 1-es szintnek, a piramis csúcsának csak egy ágyúgolyó kell. A 2-es szintnek 4 golyó kell, ebben az elrendezésben:



A 3-as szintre kilenc golyó kell, a 4-esbe tizenhat és így tovább.

Az egész piramis felépítéséhez szükséges ágyúgolyók számát az határozza meg, hogy hány szintből akarja felépíteni. Egy három szintes piramishoz 1+4+9, azaz 14 ágyúgolyó kell, a négy szinteshez 1+4+9+16, tehát 30. Ha nagyon nagy piramist akar építeni, az összegek nagyon hosszúvá és unalmassá válnak, úgyhogy esetleg arra az elhatározásra jut, hogy érdemesebb egy olyan programot írni, amely elvégzi Ön helyett a számolást. A program erre a kérdésre fog válaszolni: „Hány ágyúgolyóra lesz szükségem egy olyan gúlához, amely 'ennyi és ennyi' szintből áll?".

A program megtervezésében kulcsfontosságú az a tényező, hogy hány ágyúgolyó van szintenként. Az 1, 4, 9, 16 ... számsor ismerősnek tűnik, és hamar rájöhetünk, hogy az egyes szinteken levő golyók száma a szintek számának négyzete. A 7-es szinten például 7*7, azaz 49 ágyúgolyó kell.

És most nézzük a program részleteit. Kezdjük azzal, hoggy kigondoljuk a szükséges változókat.

Az átfogó tervnek minden szintet egyenként kell értékelnie. Kiszámíttatjuk a számítógéppel, hogy hány golyó kell az adott szintekhez, és ezt az összeget hozzáadjuk a „futó összeghez". Az elején a futó összeget nullára kell beállítani. Végül, amikor már minden szintet számításba vettünk, a futó összeg az egész piramishoz szükséges összes golyók számát mutatja. Ez jelenti a feladat megoldását.

Legyen a „futó összeg" elnevezése FS (futó szumma). Két másik változóra is szükségünk van:

1. A piramis szintjeinek száma. Ne feledjük, hogy *a programozó* nem ismeri ezt a számot; a használó feladata, hogy megadja a megfelelő értéket. Ennek a változónak SZ (szint) lehet pl. *a neve.*

2. Ahogy a program halad előre, először az 1-es szinttel, aztán a 2-es szinttel, a 3-as szinttel ... stb. foglalkozik. Szükségünk lesz egy olyan változóra is, ami azt jelöli, hogy egy bizonyos pillanatban az SZ szint közül éppen *melyikkel* foglalkozik a program. Nevezzük ezt a változót V-nek. Mivel a V felveszi az 1 és SZ közötti összes értéket, könnyen rájöhetünk, hogy ez lesz a FOR utasításban
a ciklusváltozó, így:

```
FOR V = 1 TO SZ

...........

NEXT V
```

A program változójegyzéke:

NÉV	*CÉL*
FS	Az ágyúgolyók futó összegének tárolása
SZ	A szintek száma a gúlában
V	Annak a szintnek a száma, amellyel a gép az adott pillanatban foglalkozik

Ezután leírunk néhány cselekménysort, amit a programnak el kell végeznie:

Add a V négyzetét	(Hozzáadja a V szint ágyúgolyóinak számát a futó összeghez)
PRINT FS	(Kiírja az eredményt)
Set FS = Ø	(Az FS-t nulláról indítja)
Input SZ	(Megkérdezi a használótól, hogy hány szintet akar a gúlába elhelyezni)
FOR V = 1 TO SZ	(A ciklus vezérlése minden szint figyelembevételével)
NEXT V	
STOP	

A programhoz részletekre van szükségünk, de az még ránk vár, hogy helyes sorrendbe rakjuk őket. Azt már eldöntöttük, hogy szükség van egy ciklusra, és nagy segítségünkre van, ha minden utasításról meg tudjuk mondani, hogy mikor kell végrehajtani:

a ciklus indítása előtt

vagy a cikluson belül (a ciklusmag részeként),

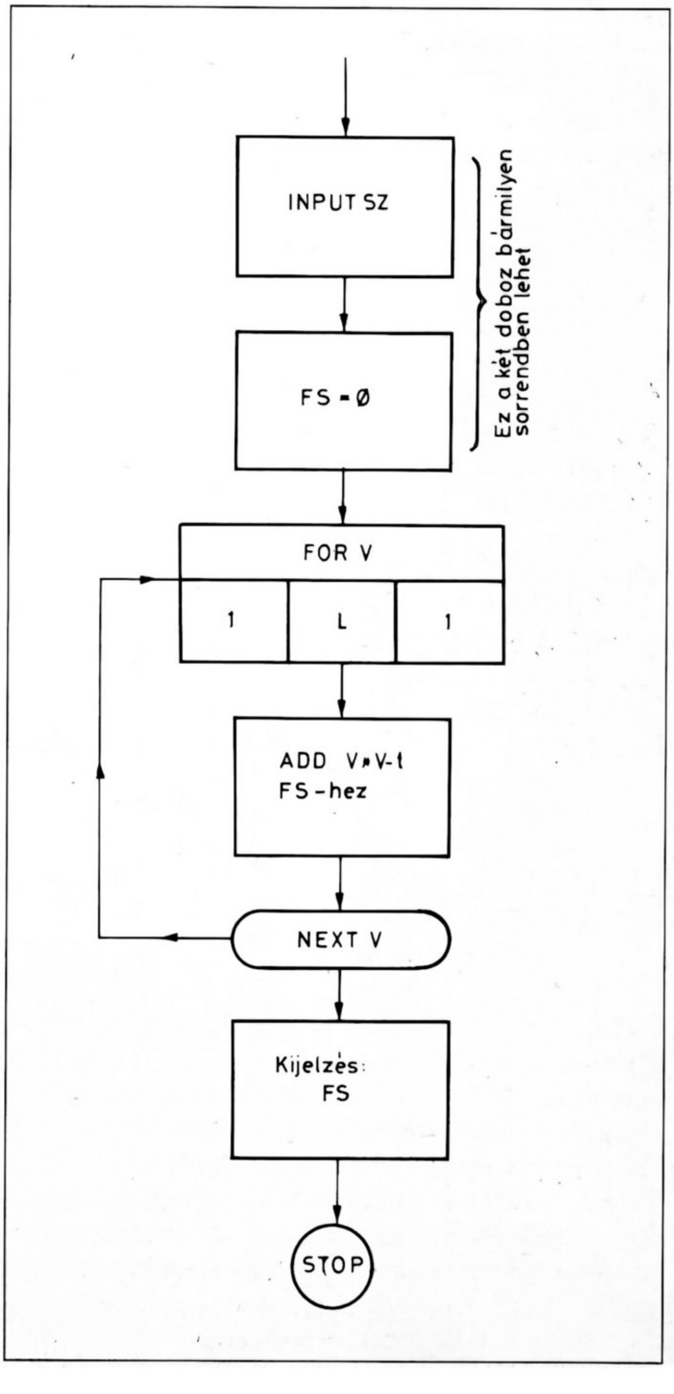
vagy a ciklus befejezése után.

Különböző érveket használhatunk. A programnak tudnia kell még a ciklus indítása *előtt,* hogy a ciklust hányszor kell majd lefuttatnia, ezért az INPUT SZ utasításnak a ciklus előtt kell lennie. Hasonlóan az FS változó értékét is a ciklus indítása előtt kell nullára állítani.

A teljes gúlához szükséges ágyúgolyók összértékében benne kell lenni minden szint értékének. Mivel azt az utasítást, hogy adja hozzá egy szint értékét FS-hez, többször meg kell ismételni, ez az utasítás a ciklus belsejébe kerül.

És végül: a számítógép nem tudja megadni a helyes választ, amíg nem vettük figyelembe az összes szintet, ezért a PRINT utasítás csak a ciklus befejezése *után* állhat.

Ennyi előkészület után megrajzolhatjuk a folyamatábrát:



A folyamatábrának megfelelő program:

```
10 INPUT "SZINTEK SZAMA"; SZ

20 FS = 0

30 FOR V = 1 TO R

40 FS = FS+V*V

50 NEXT V

60 PRINT FS; "AGYUGOLYO
        SZÜKSEGES"

70 STOP
```

Gépelje be a programot, és próbálja ki!

Nézzünk egy új feladatot. A krikett játékban egy szezonban egy játékos rendelkezhet adott számú különálló „ütési idő" felett. Minden egyes időben felírhat (nyer) néhány „futás"-t: Ha jó játékos, és szerencséje van, sokat, ha nem túl ügyes, csak keveset (vagy egyet sem). Ha meg akarja tudni, hogy valaki milyen eredménnyel játszott az egész évadban, számítsa ki a futások átlagát, ütési időnként. Az átlagot úgy kapjuk meg, hogy összeadjuk az összes nyert futás számát, és elosztjuk az egész évad ütési idő számával. Ha például háromszor játszik, és az eredménye 20, 30 és 70, az *átlaga* (20+30+70)/3, azaz 40 futás ütési időnként.

Tekintsünk egy programot, amely elvégzi ezt a számítást. Meg kell kérdezni az ütési idők számát és az azokhoz tartozó eredményt, hogy össze tudjuk őket adni. A képernyőn a következők jelenhetnek meg:

```
RUN

ÜTÉSI IDŐK SZÁMA? 3

EREDMÉNY? 20

EREDMÉNY? 30

EREDMÉNY? 70

ÁTLAG = 40
```



Készítse el a programot a feladat megoldására. Segítségül megadjuk a változójegyzéket és az összes utasítást, de a sorrendjüket összekevertük, és nincs sorszámuk. Először rajzolja meg a vázat a ciklusvezérlő utasításokkal, aztán illessze be a többi utasítást a megfelelő helyre. Végül futtassa le a programot a számítógépen és ellenőrizze, hogy működik-e. Ha teljesen elakad, keresse meg a helyes választ a B. Függelékben, de ne feledje: ez a kudarc beismerése!

NÉV	*CÉL*
J	Az évad ütési alkalmainak száma
Q	A ciklusváltozó
RS	Arra használjuk, hogy összeadja az összes sikeres ütést
S	Az egyes ütési alkalmak találatai

```
NEXT Q

INPUT "UTESI IDOK SZAMA"; J

INPUT "EREDMENY"; S

PRINT "ATLAG = ";RS/J

STOP

RS = 0

FOR Q = 1 TO J

RS = RS+S
```

Vége a 12.1 Gyakorlatnak

12.2 Gyakorlat

Egy olyan feladattal fejezzük be ezt a leckét, amelyet minden segítség nélkül, Önnek kell megoldania. Ha elmegy a Postahivatalba, elég gyakran megesik, hogy órákig áll sorban valaki miatt, aki százával veszi a bélyegeket. Hallható, ahogy sorolja: „Nyolcvanhármat a 12 pennysből, száztizenhetet a 16 pennysből és harmincötöt a 75 pennysből,...” és így tovább. Amikor minden bélyeg a pulton van, a postai alkalmazott hosszan számolgatja, hogy mennyit kell érte fizetni.

Írjon egy olyan programot, amely segít az alkalmazottnak. Az eredmény ilyesmi lesz:

```
RUN
TÉTELEK SZÁMA?                (4)
1TÉTEL
   BÉLYEGEK SZÁMA?            (2Ø)
   ÉRTÉKE (EGYNEK)?           (15)
2TÉTEL
   BÉLYEGEK SZÁMA?            (5)
   ÉRTÉKE (EGYNEK)?           (75)
3TÉTEL
   BÉLYEGEK SZÁMA?            (4)
   ÉRTÉKE (EGYNEK)?           (1)
4TÉTEL
   BÉLYEGEK SZÁMA?            (1ØØ)
   ÉRTÉKE (EGYNEK)?           (14)
TELJES ÖSSZEG=2Ø79 PENNY
```

A számokat a felhasználó írja be

A programnak 1Ø utasításból kell állnia, beleértve a STOP-ot is. Ebből négy alkotja a ciklusmagot, amelyet minden tételnél egyszer végrehajt a gép. Addig mégse kezdje el a programot megírni, amíg nem készítette el a hozzá való tervet, változójegyzéket és folyamatábrát. Szánjon rá elegendő időt!

Ha a kellő energiabefektetés ellenére sem tudja megoldani a feladatot, lapozzon vissza egy néhány fejezetnyit oda, ahol biztonságosan eligazodik, és onnan kezdve vegye át újra az egész anyagot.

Végül hasonlítsa össze a saját megoldását a B. Függelékben lévővel.

Vége a 12.2 Gyakorlatnak

A 12. Leckéhez tartozó önellenőrző kvíz neve LECKE12KERDESEK.

13. LECKE

13.1 Gyakorlat

Ebben a leckében olyan témáról lesz szó, amely könnyű is, szórakoztató is: a számítógépes hangkeltésről és a zenélésről.

Töltse be a HANG DEMO nevű programot, tekerje fel a tv-n a hangerőt, és játssza végig a programban szereplő hanghatásokat. A hatások széles skálájúak, és máris érzékelheti, hogy a hangkeltés terén mire képes a számítógép.

Megfelelő időben meg akarja majd tervezni és beprogramozni a saját szerzeményét; ezzel foglalkozik a teljes lecke.

A számítógépen két parancs vezérli a hangkeltést: a VOL ('volume' = hangerő) és a SOUND (= hang).

Kezdjük a hangerővel, mivel ez az egyszerűbb a kettő közül. A programban a VOL kulcsszó után egy szám (vagy kifejezés) áll, melynek értéke Ø-tól 8-ig terjedhet. Ez a parancs vezérli a számítógép által keltett hangok erősségét; hatása a 'hangerő szabályozás'. Akkor szólnak a hangok a leghangosabban, ha a

VOL 8

-at adjuk meg, és akkor halkulnak le a teljes csendig, ha a

VOL Ø

-t állítjuk be.

A közöttük levő hangerő-fokozatokat a VOL 2, VOL 4 stb-vel lehet választani.

Hangkeltést alkalmazó programokban mindig a program eleje körül kell állnia a

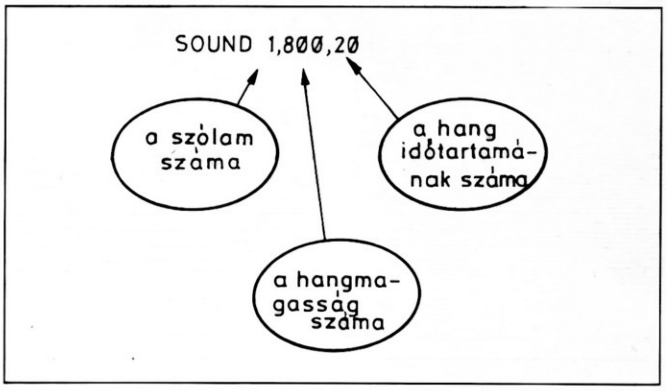
VOL 7

-nek. Amikor a programot futtatjuk, a tv-készülék vagy monitor hangerejét is fel kell csavarni, mert különben nem hallunk semmit.

A SOUND paranccsal lehet egy adott zenei hangot kelteni. Minden hangnak két fontos tulajdonsága van:

– a hangmagasság (mély vagy magas),
– a hang időtartama (a hangzás hossza).

A SOUND parancsban a kulcsszót három ilyen szám vagy kifejezés követi:



A számítógép három különböző szólammal rendelkezik, amelyek 1, 2 és 3-mal vannak számozva. Pillanatnyilag csak az 1-es számú szólamot használjuk. Minden hang parancs így fog kezdődni:

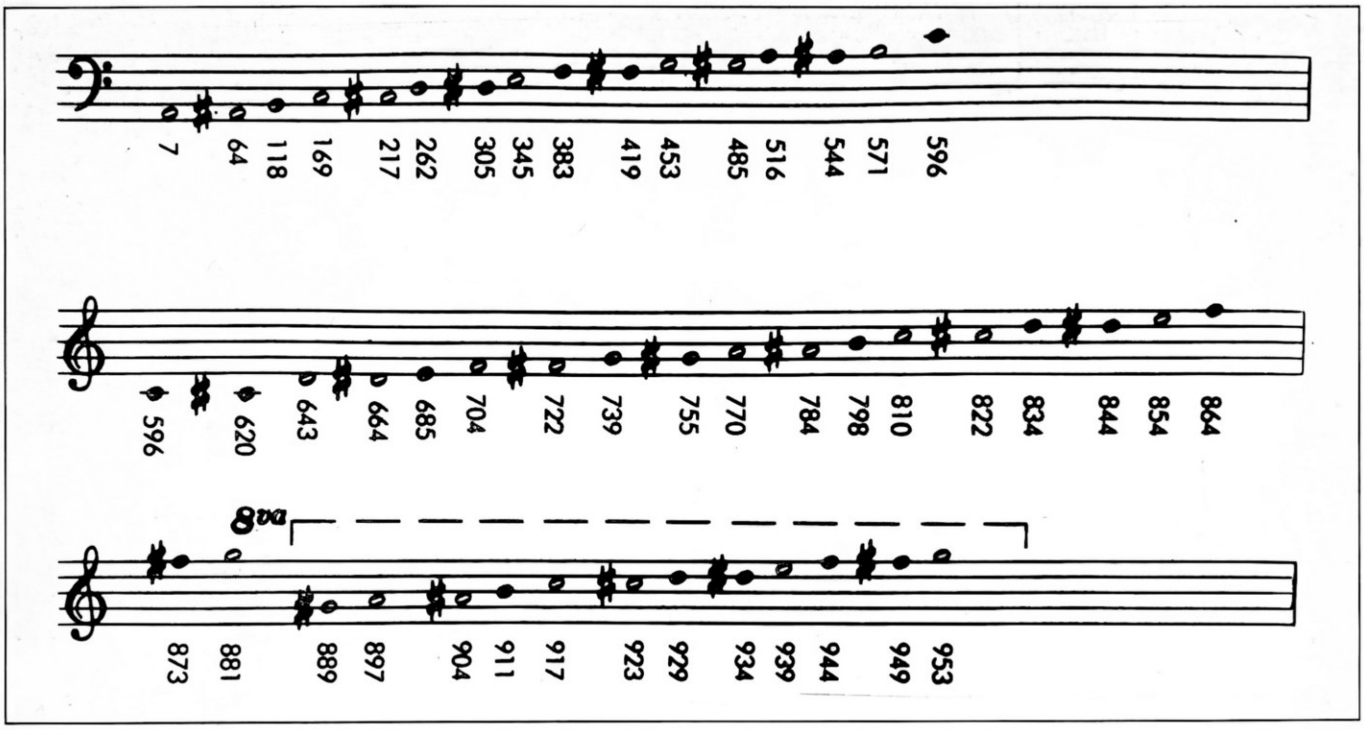
SOUND 1, ...

A hangmagasságot egy szám vezérli, amelynek értéke Ø-tól 1Ø23-ig terjedhet. Nagyjából azt mondhatjuk, hogy minél nagyobb a szám értéke, annál magasabb hangot vált ki, de a hangok közötti távolság egyáltalán nem egyenletes. Ez a rövid program a Ø és 1Ø23 közötti összes lehetséges hangmagasságot lejátssza, és a gép hangterjedelmének határait is érzékeltethetjük vele.

```
1Ø VOL 7
2Ø FOR J=Ø TO 1Ø23
3Ø SOUND 1,J,1
4Ø NEXT J
```

A következő ábra megmutatja, hogy a hangmagasságot jelölő számok hogy viszonyulnak a zenei skála hangjegyeihez. Ez az információ hasznos lesz, amikor zenét ír a gépre.

89



A hangzás hosszúságát 1/60 másodperces egységekben az időtartam száma állítja be. Így például a SOUND 1, 9ØØ, 12Ø utasítással megadott hang pontosan két másodpercig fog hangzani.

A hang lehetőséget legegyszerűbben és legpraktikusabban arra tudjuk használni, hogy közöljük a használóval: a begépelt válasza elfogadható-e vagy sem. Tekintsünk egy olyan programot, amelyik megkérdezi valakitől, hogy mit inna: vörös bort (V), fehér bort (F), sört (S) vagy esetleg semmit (N). A használót arra kérjük, hogy a választásának megfelelő betűt gépelje be. Ha azt gépeli be – amint kell –, hogy V, F, S vagy N, a gép egy vidám 'bip' hangot hallat, és továbbmegy a következő kérdésre. Ha a használó a fentiektől eltérő, más betűt ír be, a gép egy bánatos, mély hangot hallat és újra felsorolja a választékot. Egy ilyen sorozatot így lehet leírni:

```
1Ø VOL 7

2Ø PRINT "VOROS BOR" (V), FEHER
   BOR (F)"

3Ø PRINT "SOR (S) VAGY SEMMI
   (N)"

4Ø INPUT "SZIVESKEDJEN MEGADNI
   A MEGADOTT BETUT"; L$

5Ø IF L$="V" THEN 14Ø

6Ø IF L$="F" THEN 14Ø

7Ø IF L$="S" THEN 14Ø

8Ø IF L$="N" THEN 14Ø

9Ø PRINT

1ØØ PRINT "KEREM, VALASZA LEGYEN
    V, F, S VAGY N"

11Ø PRINT

12Ø SOUND 1,2Ø,7Ø:REM MELY HANG*

13Ø GOTO 2Ø
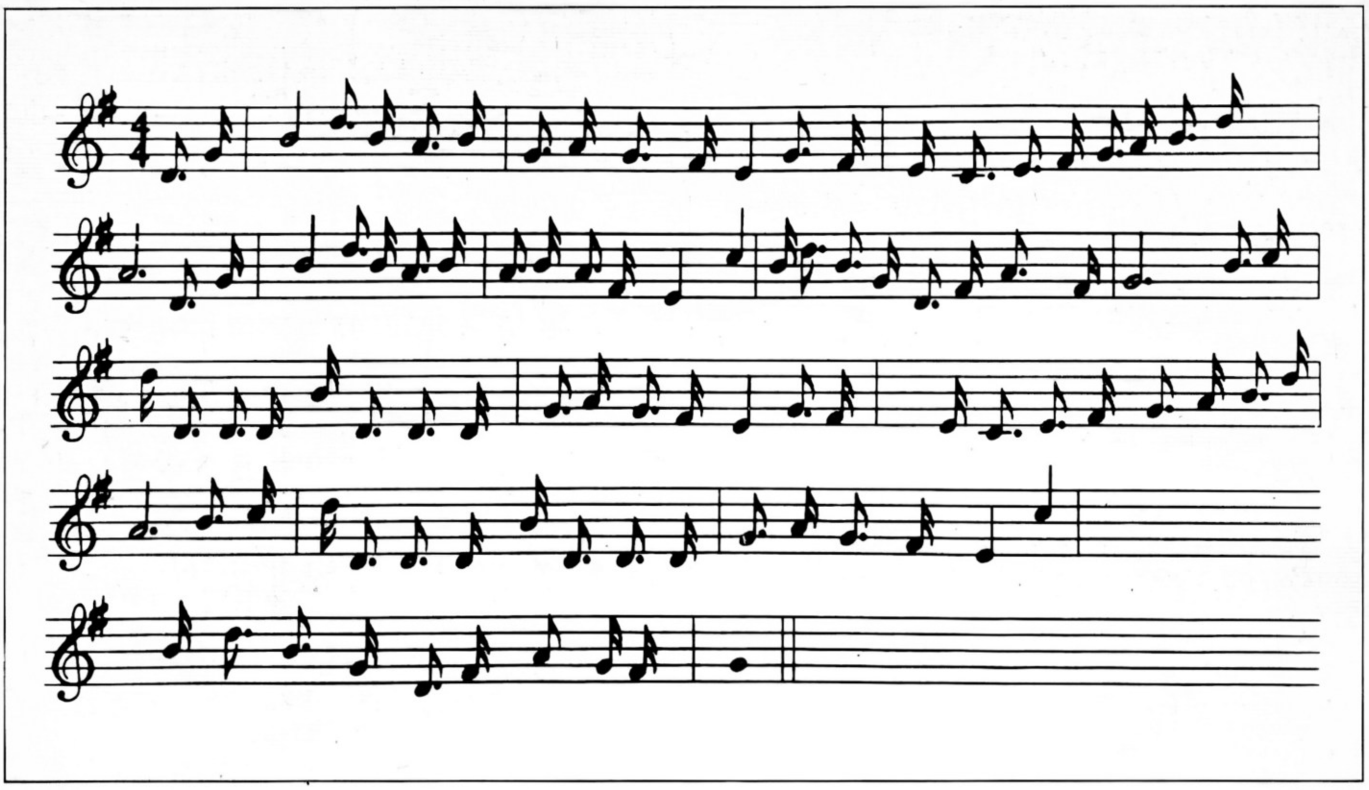
14Ø SOUND 1,93Ø,12:REM BIP

15Ø PRINT "RENDBEN"

16Ø ...
```

A SOUND parancsok arra is használhatók, hogy a számítógéppel dallamokat játszassunk el. A dallam minden hangjegyét a saját megfelelő SOUND parancsa játssza el. Íme egy ismert skót dal, *„Úton a szigetekhez"*.

* A kettőspont egy új utasítást vezet be ugyanazon sorban *(a lektor megjegyzése)*.



A DALLAM nevű programban tároltuk kódolva a dallam első részét, ezt csak be kell tölteni és futtatni. Ha a programot kilistáztatja, látni fogja, hogyan készült. Minden hangjegye egy SOUND utasításra lett lefordítva.
A dallam minden üteme 32 hatvanadmásodperc hosszú (ez megfelel 60*60/32, vagyis 112 metronómütemnek percenként).

Az első hangjegy egy D hangmagasságú pontozott nyolcad (vagyis három negyed), ezért a kódja

SOUND 1, 643 24

D magasság 32 háromnegyede

Az egyik leggyakrabban használt zenei fogás a szünet. Normális körülmények között a gép bármilyen hosszú szünetet tud tartani, ha egy ilyen üres ciklust kap:

```
FOR J = 1 TO 1ØØØ: NEXT J
```

A zenében azonban nem ilyen egyszerű a dolog. Törölje a tárat, és írja be:

```
1Ø VOL 7

2Ø SOUND 1, 4ØØ, 12Ø

3Ø FOR J = TO 1ØØØ: NEXT J

4Ø SOUND 1, 4ØØ, 12Ø
```

Amikor a programot lefuttatja, azt tapasztalja, hogy nincs szünet ott, ahol várta! Miért? Mert a SOUND utasítás pusztán csak beindítja a hangot. Míg a hang továbbra is szól, a gép végrehajtja a következő utasításokat. Csak akkor kerül várakozási állapotba, ha egy újabb SOUND utasítással találkozik, mielőtt az előző hang elhallgatott volna.

Most már pontosan tudjuk, mi történt, miközben a program futott. A gép belekezdett az első hangba, majd rögtön továbbment a 3Ø. sorban levő üres ciklusra. Egy másodperc múlva befejezte a ciklust, és megpróbálta beindítani a második hangot – de az első még mindig szólt. Amint az első hang elhallgatott, a második azonnal megszólalt, szünet nélkül.

Most már világos a probléma megoldása. Ha azt akarjuk, hogy egy hangot szünet kövessen, az üres ciklusnak végig ki kell tartania a hangzás és a csend alatt is. Tehát ha egy másodpercig szóló hangot egy másodpercnyi szünettel akarunk folytatni, a ciklusnak két másodperc számítógép időt kell tartalmaznia.

Hasonló problémával kerülünk szembe, amikor két azonos hangot akarunk lejátszani egymás után. A

```
SOUND 1, 4ØØ, 3Ø

SOUND 1, 4ØØ, 3Ø
```

sorozat hatása megkülönböztethetetlen a következő utasítás hatásától:

```
SOUND 1, 400, 60
```

mert a két hang egymást követi szünet nélkül. Ezért így kell összeállítani:

```
SOUND 1,400,30

FOR J=TO 500: NEXT J*

SOUND 1,400,30
```


Gyakorlásként fejezze be a skót népdal kódolását. Fordítson külön figyelmet az ismételt hangokra.

Vége a 13.1 Gyakorlatnak

* Az üres FOR ciklus akkor tart 1 másodpercig, ha a végérték kb. 770 (nem teljesen lineáris az összefüggés). *(A lektor megjegyzése.)*

13.2 Gyakorlat

A legrövidebb hang, amelyet a számítógéppel keltetni tudunk, csak egy hatvanadmásodperc hosszú. Ezért nagyon érdekes hanghatásokat érhetünk el, ha sok ilyen rövid, különböző magasságú hangot játszunk le. A következő példa ezt mutatja:

```
10 VOL 7
20 FOR X=700 TO 900 STEP 4
30 SOUND 1,X,1
40 NEXT X
50 GOTO 20
```

Más hatást érünk el, ha így kibővítjük:

```
35 SOUND 1,X=50,1
```

Most nézzük meg a számítógép két másik szólamát. A 2. szólam önmagában épp úgy viselkedik, mint az 1. szólam. Ha megpróbáljuk a kettőt egyszerre használni, erős elektronikus interferencia alakul ki köztük – a hatás érdekes, de harmonikusnak aligha nevezhető! A 3. szólamot arra használjuk, hogy a gyakran 'fehér zajnak' nevezett hangzást keltsük vele: ez egy nehezen meghatározható hangmagasságú sistergés. Ezt a hangzást ragyogóan lehet alkalmazni gőzgépek, repülőgépek, rakéták és űrhajók hangjának szimulálására, ha tudja, hogy mihez hasonlít hangjuk. Próbálja ki a következő programot:

```
10 VOL 7
20 FOR X=850 TO 600 STEP–1
30 SOUND 3,X,2
40 NEXT X
```

és egy bonyolultabbat:

```
10 VOL 7
20 J=50
30 FOR L=1 TO 100
40 J=0.97*3
50 M=J+15
60 SOUND 3,860–M,M/3
70 FOR K=1 TO 4*M:NEXT K
80 NEXT L
```

Vége a 13.2 Gyakorlatnak

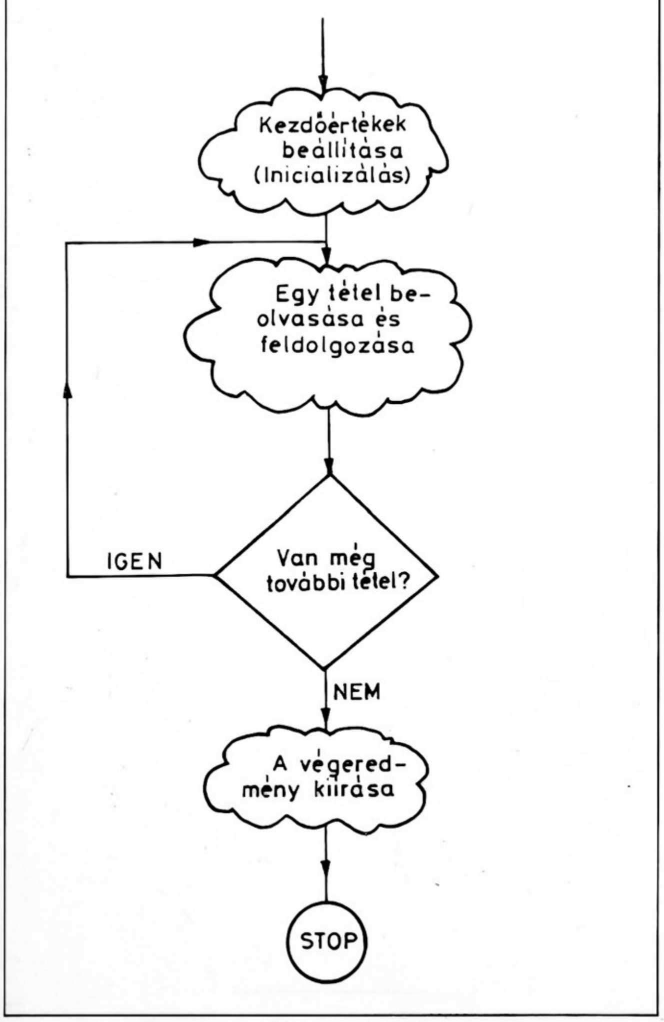
14. LECKE

14.1 GYAKORLAT

14.2 GYAKORLAT

Ebben a leckében a számítógép egy rendkívül gyakran használt alkalmazási területével foglalkozunk: a gépbe bevitt számos, különböző tételből álló információt feldolgoztatjuk és az összesített végeredményt kiíratjuk. Ha például valaki nyomon akarja követni bankszámlája alakulását, betáplálhatja a kitöltött csekkjeinek részletes adatait és a bankszámlájára befizetett összegeket, és a gép megmondja a hét végi egyenleget. Vagy vegyünk egy másik példát: egy tanár beírhatja a számítógépbe az összes osztályzatot, amit a tanulók az osztályban kaptak, a gép pedig megadja az osztályzatok átlagát.

Minden ilyen típusú program ugyanazt az alapsémát követi, amelyet így lehet egy folyamatábrán összefoglalni:



Bemutatunk egy nagyon egyszerű példaprogramot, amely beolvas 1Ø számot és megadja átlagukat:

```
1Ø S=Ø
2Ø P=1
3Ø INPUT X          } A tétel beolvasása és
4Ø S=S+X            } feldolgozása
5Ø P=P+1            } Van további
6Ø IF P < 11 THEN 3Ø } tétel
7Ø PRINT "ATLAG="; S/1Ø  Az eredmény
                         kiírása
8Ø STOP
```

Változójegyzék:

S: A tételek értékének összeadására használjuk
P: A tételek számlálására használjuk
X: Az egyedi tételek bevitelére használjuk

Ha nem érti a program működését, végezzen nyomkövetést a 3, 6, 2, 7, Ø, 9, 8, 3, 12, 1Ø input értékekkel.

Hogy a program szerkezete világosabb legyen, ebben a példában egy IF-THEN utasítást használtunk a ciklus vezérlésére. A gyakorlatban FOR...NEXT-tel írnánk meg a programot:

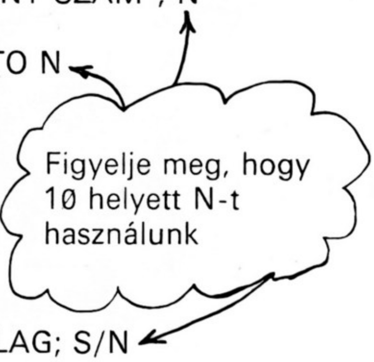
```
1Ø S=Ø
2Ø FOR P=1 TO 1Ø
3Ø INPUT X
4Ø S=S+X
5Ø NEXT P
6Ø PRINT "ATLAG";S/1Ø
7Ø STOP
```

Vizsgáljuk meg a programnak azt a részét, amelyik azt kérdezi: „Van további tétel?”. Az

első példában a kérdést egyszerű számlálással, és a P<11 feltétellel oldottuk meg, amely mindaddig igaz volt, amíg a tizedik tételt be nem olvastuk és hozzá nem adtuk a végösszeghez. Ehhez a módszerhez a programozónak előre tudnia kell, hogy hány tétellel lesz dolga. Gyakorlatilag ez a módszer használhatatlan, mert nagyon rugalmatlan: valahányszor megváltozik a tételek száma, mindannyiszor új programot kellene írnunk az átlag kiszámítására.

Sokkal jobb programot írhatunk, ha abból indulunk ki, hogy *a használó* meg tudja mondani a számítógépnek, hogy hány tételt akar feldolgozni. A következő program bármilyen tételszámra érvényes:

```
10 S=0
20 INPUT "HANY SZAM"; N
30 FOR P=1 TO N
40 INPUT X
50 S=S+X
60 NEXT P
70 PRINT "ATLAG; S/N
80 STOP
```



Változójegyzék:

S: A tételek értékének összeadására használjuk
P: A tételek számlálására használjuk
X: Az egyedi tételek bevitelére használjuk
N: A tételek *számának* megőrzésére használjuk

Eddig ismerős talajon jártunk, de mi történik abban az esetben, amikor a használónak rengeteg tételszámot (ezret vagy még többet) kell betáplálnia? Azt nem kívánhatjuk tőle, hogy előre számolja meg a tételeket, és nem valószínű, hogy hibátlanul végezné el a számolást.

A ciklusvezérlés másik módja az, hogy egyáltalán nem használunk egy előre meghatározott számot, csak azt közöljük a számítógéppel, hogy mikor értek véget a tételek. Megkérhetnénk például arra a használót, hogy minden ciklusban válaszoljon a kérdésre: van még adat? Ez egy ilyen programhoz vezetne:

```
10 S=0
20 N=0
30 INPUT "KOVETKEZO SZAM";X
40 S=S+X
50 N=N+1
60 INPUT "VAN MEG SZAM";M$
70 IF M$="IGEN" THEN 30
80 PRINT "AZ ATLAG:";S/N
90 STOP
```

Változójegyzék:

S: A tételek összeadására használjuk
N: A tételek számlálására használjuk
X: Az egyedi tételek bevitelére használjuk
M$: A „Van még szám” kérdésre adott válasz tárolására használjuk

Ha lefuttatná ezt a programot, ilyen lehetne a kijelzés:

```
KÖVETKEZŐ SZÁM? [4]
VAN MÉG SZÁM? [IGEN]
KÖVETKEZŐ SZÁM? [7]
VAN MÉG SZÁM? [IGEN]
KÖVETKEZŐ SZÁM? [10]
VAN MÉG SZÁM? [NEM]
AZ ÁTLAG: 7
BREAK IN 90
READY
```



Világosan látható ennek a sémának a fogyatékossága. A szerencsétlen használónak minden szám után (az utolsó kivételével) be kell gépelnie, hogy IGEN. Ez kétszer annyi időt vesz igénybe, és megkétszerezi a hibalehetőségek számát.

Sokkal jobb módszer, ha a *végjel* vagy *terminátor* nevű speciális értékkel jelöljük meg a tételek beáramlásának végét.
A terminátornak lehetőleg olyan értéket kell választanunk, amely nem fordulhat elő a tételek között. Ha például a futballban lőtt gólok átlagát szándékozunk a programmal kiszámítani, nyugodtan használhatjuk a 1ØØØØØØ-t, mert abban biztosak lehetünk, hogy nincs olyan csapat a világon, amelyik egy meccsen egy millió gólt lőne.

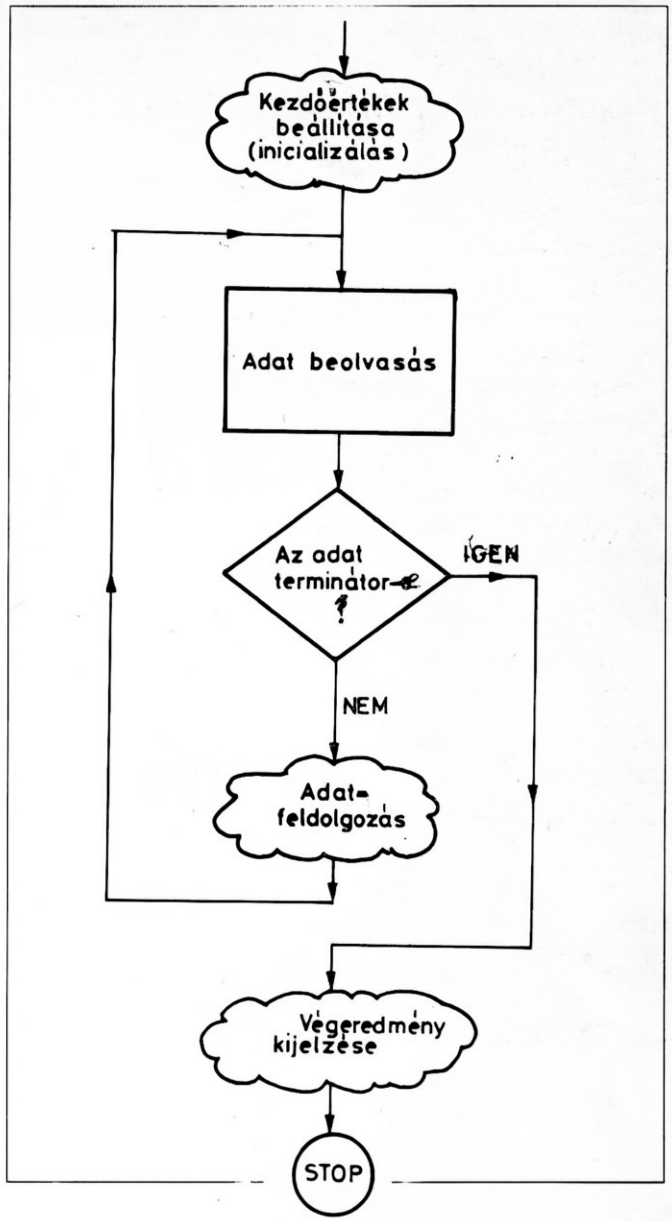
Ha ebben a szellemben írunk egy programot, a képernyőn valami ilyesmi látható:

```
HASZNALJ 1ØØØØØØ-t AZ INPUT
LEZARASARA

KOVETKEZO SZAM?   5
KOVETKEZO SZAM?   7
KOVETKEZO SZAM?   Ø
KOVETKEZO SZAM?   2
KOVETKEZO SZAM?   1
KOVETKEZO SZAM?   1ØØØØØØ
AZ ATLAG: 3
BREAK...
```

A használó gépeli be

Ha ezt a rendszert akarjuk használni, át kell rendeznünk az átfogó folyamatábrát; nevezetesen a „Van még adat?" kérdésnek az adatokat egyenként feldolgozó blokk elé kell kerülnie, ugyanis egyébként a gép a végjelet úgy kezelné, mint bármelyik más tételt, és elrontaná az eredményt.



Ennek alapján egy átlagot kereső program meglehetősen nyilvánvalóan a következő:

```
10 PRINT "JELEZZE 1ØØØØØØ-EL"
20 PRINT "AZ ADATOK VEGET"
30 S = Ø
40 N = Ø
50 INPUT "KOVETKEZO SZAM";X
60 IF X = 1ØØØØØØ THEN 100
70 S = S+X
80 N = N+1
90 GOTO 50
100 PRINT "ATLAG=";S/N
110 STOP
```

Változójegyzék:

S: A tételek értékének összeadására használjuk
N: A tételek számlálására használjuk
X: Az egyedi tételek bevitelére használjuk

Összefoglalásul elmondhatjuk hogy négy olyan módszert ismertünk meg, amellyel a program tudomására hozhatjuk, hogy hány adatot kell beolvasnia.

1. A programozó határozza meg a tételek számát. Csak kezdők alkalmazzák, a gyakorlatban használhatatlan.

2. Az adatok számát a használó előre meghatározza. 2Ø vagy kevesebb tétel esetén jól használható módszer.

3. A használó megjelöli *minden egyes tétel után,* hogy van-e még több. Elviselhetetlenül unalmas.

4. Az adatáramlást egy speciális érték zárja le. Jó módszer, a többinél általában jobb.

14.1 Gyakorlat

Írjon egy egyszerű programot, amely a bankszámlája egyenlegét dolgozza fel. Az input legyen a régi egyenleg és a kitöltött csekkek részletezése, amiből kiszámolható a jelenlegi egyenleg vagy hiteltúllépés. Terminátornak nullát használjunk, mert nulla £-os csekket úgysem ír soha senki. Most a követelésekkel ne törődjünk.

Úgy tervezze meg a programot, hogy a képernyőn az alábbiak jelenjenek meg:

a) RÉGI EGYENLEG? 5.24

RÉSZLETEZZE A CSEKKET,

VÉGJELKÉNT Ø-T

HASZNÁLJON

ÖSSZEG? 1.73

ÖSSZEG? 2.00

ÖSSZEG? Ø

AZ ÖN EGYENLEGE 1.51 £

(A használó írja be)

b) RÉGI EGYENLEG? 4.Ø2

RÉSZLETEZZE A

CSEKKET, VÉGJELKÉNT

Ø-T HASZNÁLJON

LEGYEN

ÖSSZEG? 3.5Ø

ÖSSZEG? 1.5Ø

ÖSSZEG? 3.ØØ

ÖSSZEG? Ø

AZ ÖN HITELTÚLLÉPÉSE 3.98 £

(A használó írja be)

Hasznos tanács: Az eredmény kiírása most egy kicsit bonyolultabb lesz, mint máskor. Az aktuális egyenleget tartalmazó változó, B értéke *negatív* lesz (azaz Ø-nál kisebb), ha

túlléptük betétünket. A programban ezt a B<Ø feltétellel ellenőrizhetjük.
A megoldáshoz készítsen folyamatábrát és változójegyzéket is. A megoldás helyességét ellenőrizze a B. Függelékben.

Vége a 14.1 Gyakorlatnak

Bizonyos feladatoknál az információfolyam különböző tételeit különbözőképpen kell kezelni. Az ilyen feladatokat megoldó programok fő ciklusában általában IF utasítás található. Tegyük fel például, hogy egy szerencsejátékban egy nagyon rossz sorozat után arra kezd gyanakodni, hogy az érmében eltolódott a súlypont, mert sokkal többször jön ki 'fej', mint 'írás'. A gyanúját úgy tudja igazolni, hogy sokszor dob egymás után, és feljegyzi a kijött 'fej' és 'írás' darabszámait. Ha akarja, a számítógép segít a találatok feljegyzésében, tehát egy olyan programot szeretne írni, amely ilyen jellegű kijelzést ad:

```
IRJ F-ET HA FEJ VAN

I-T HA IRAS VAN

V-T HA VEGE

KÖVETKEZŐ DOBÁS? F

KÖVETKEZŐ DOBÁS? F

KÖVETKEZŐ DOBÁS? I

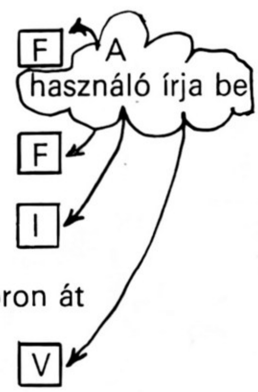
... és így tovább, 547 soron át

KÖVETKEZŐ DOBÁS? V

547 DOBÁSBÓL 49Ø VOLT FEJ

ÉS 57 IRÁS

READY
```



Ebből már levonhatjuk a következtetést az érme szabálytalanságáról.

Tervezzük meg, és írjuk meg ezt a programot, kezdve a változójegyzékkel és a folyamatábrával egészen a BASIC utasításokig.

A minta kijelzés mutatja, hogy egy speciális értéket, a V-t használhatjuk az adatsor lezárására. A folyamatábra olyan lesz, mint a 14. lecke második ábrája, tehát csak meg kell növelnünk a felhőket. A programhoz nyilvánvalóan 3 változóra van szükség:

F: A 'fej' dobások számlálására I: Az 'írás' dobások számlálására I$: Adatbevitelre

(Az F és I nevek itt is szabadon variálhatók.)

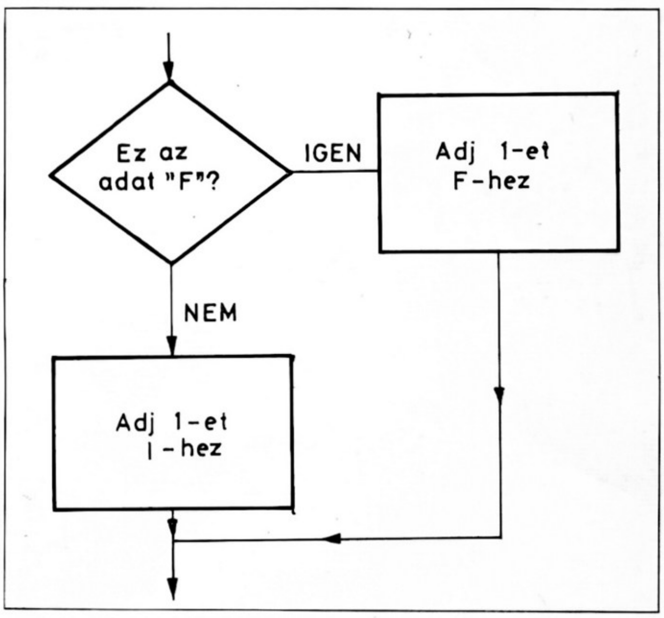
Lehet, hogy valakinek kedve támad még egy negyedik változó beillesztésére is, amely az összes dobásokat számlálná, de ennek nem sok értelme van: a végösszeget az F+I kifejezés (a 'fej' *plusz* az 'írás' dobások) adja meg.

Következő lépésként a kezdőértékek beállításával foglalkozzunk. Két teendő van:

– Az F és I változók értékének nullára állítása.

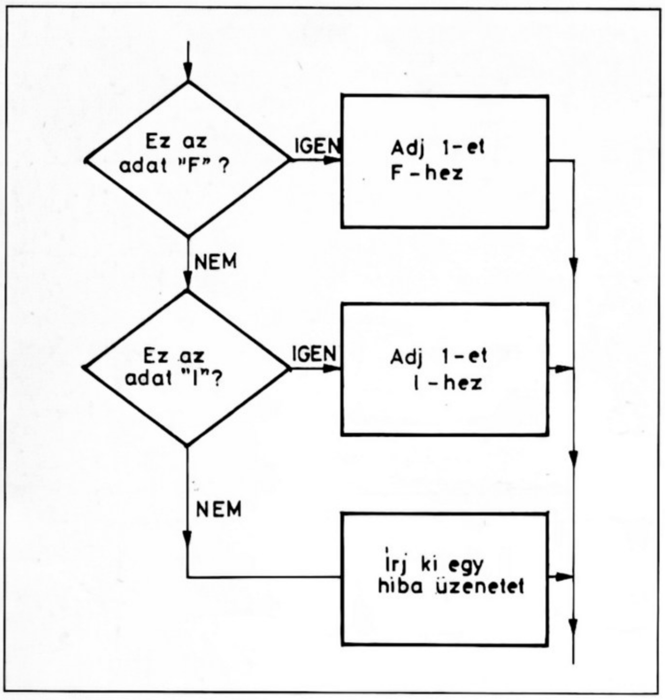
– A fejléc üzenet kijelzése.

Most a felhő belsejében levő fő ciklus következik, amely az adatokat egyenként feldolgozza. E szinten a V-t már kiszűrtük, az adatoknak vagy F-nek, vagy I-nek kell lenniük. A felhőnek az az alapvető feladata, hogy hozzáadjon egyet a 'fej' vagy az 'írás' végösszegéhez. Az egyik lehetséges megközelítés:
„Ez F? Ha nem, akkor I-nek kell lennie." Ezt a gondolkodást így lehetne folyamatábrán bemutatni:



Az az igazság, hogy egy gyakorlott programozó soha nem használná ezt a módszert, mert így a program nincs felkészítve a gépelési hibák kiszűrésére. Ha a használó F helyett valamelyik mellette levő billentyűt üti le, a gép I-nek számolná, holott biztosan nem ez volt a célja.

Sokkal jobb, ha a hibalehetőségekkel is számolunk:



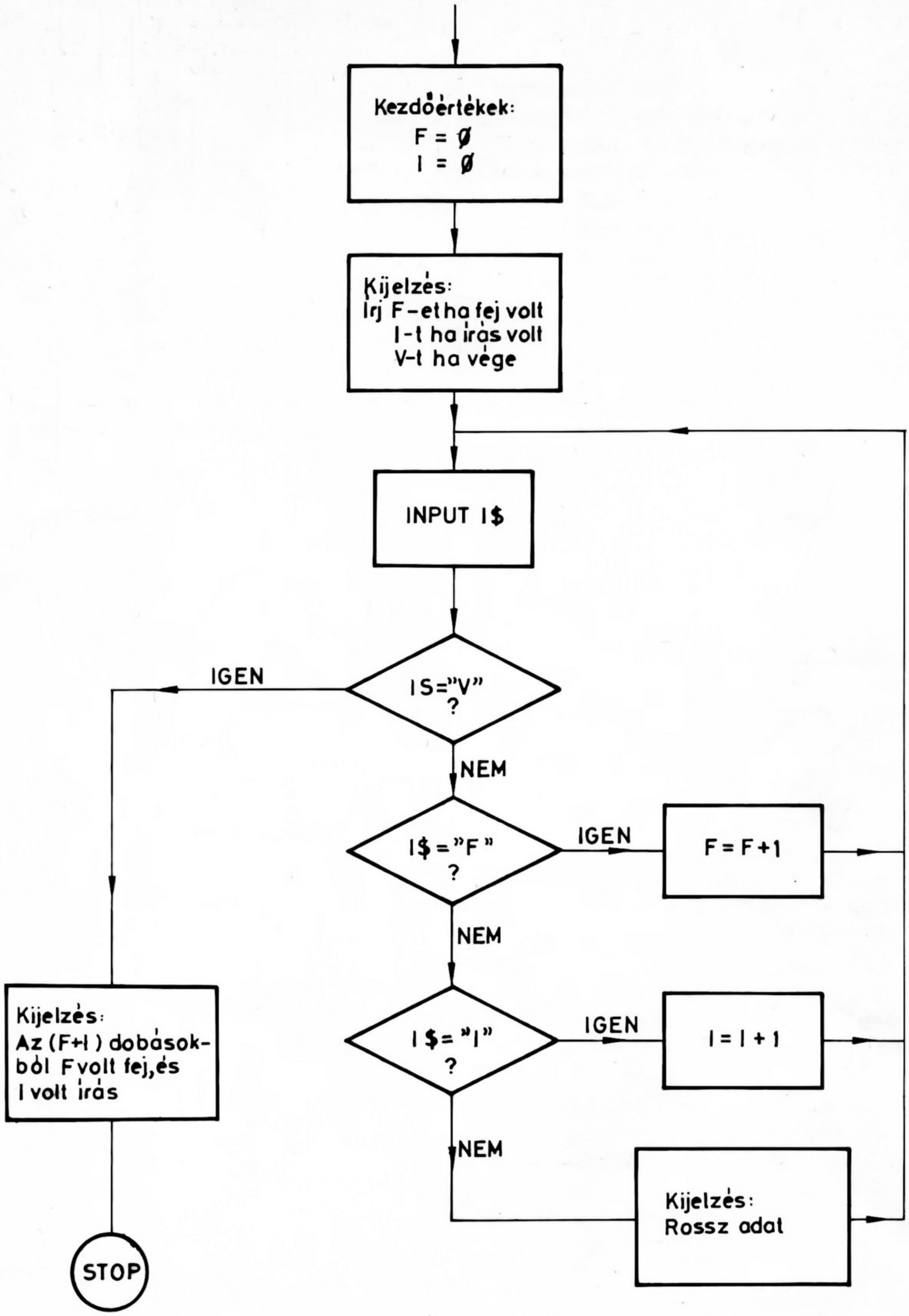
Azt a programot nevezzük *megbízhatónak,* amely feltételezi, hogy a használó véthet hibát, de nem engedi, hogy végzetes következményekkel járjon.

Végül kibővíthetjük az „összegzés” felhőt olyanra, hogy megjelentesse a háromsoros választ a kijelzés végén.

A következő oldalon láthatjuk a kibővített folyamatábrát.

A folyamatábrának megfelelő programot is megadjuk. Figyelje meg, hogy a fő ciklus BASIC megfelelője egy kissé kusza. Ez sajnos elkerülhetetlen, valahányszor egy kétdimenziós folyamatábrát kell utasítások egyetlen sorozatával leírnunk.

```
 1Ø F = Ø
 2Ø I = Ø
 3Ø PRINT "IRJ F-ET HA FEJ VOLT"
 4Ø PRINT "I-T HA IRAS VOLT"
 5Ø PRINT "V-T HA VEGE"
 6Ø INPUT "KOVETKEZO DOBAS"; I$
 7Ø IF I$ = "V" THEN 16Ø
 8Ø IF I$ = "F" THEN 12Ø
 9Ø IF I$ = "I" THEN 14Ø
1ØØ PRINT "ROSSZ ADAT"
11Ø GOTO 6Ø
12Ø F = F+1
13Ø GOTO 6Ø
14Ø I = I+1
15Ø GOTO 6Ø
16Ø PRINT "F+I;"SZAMU DOBASBOL"
17Ø PRINT F;"FEJ"
18Ø PRINT "ES";I;"IRAS VOLT"
19Ø STOP
```


Kezdőértékek:
F = Ø
I = Ø
Kijelzés:
Irj F-et ha fej volt
I-t ha írás volt
V-t ha vége
INPUT I$
IGEN
I$="V"
?
NEM
I$ = "F"
?
IGEN
F = F+1
NEM
Kijelzés:
Az (F+I) dobásokból F volt fej, és
I volt írás
I$ = "I"
?
IGEN
I = I + 1
NEM
Kijelzés:
Rossz adat
STOP

14.2 Gyakorlat

a) Ha egy program túl sok adat bevitelét igényli, lehet, hogy gépelés közben a használó már nem nézi a képernyőt. Mint ahogy a 13. leckében elmondtuk, jó ötlet olyan programot készíteni, amely hangokkal és kijelzett üzenetekkel reagál. Beállíthatunk például egy 'bip' hangot az elfogadható adatokra és egy durva zörejt az elfogadhatatlanokra.

Nézzük meg a fej vagy írás programot. Amikor a használó F-et gépel be, a gép mindig végrehajtja a 12Ø és 13Ø sorok utasításait.

A következő kiegészítéssel megszólaltatjuk a megfelelő hangot:

```
123 VOL 7

125 SOUND 1,9ØØ,1
```

Töltsük be a kazettáról a FEJEK programot (így rövidebb, mintha magunk gépelnénk be), és szerkesszük meg úgy, hogy minden inputra – attól függően, hogy az adat helyes vagy sem – megfelelő hanggal válaszoljon.

b) Valamikor az órákra fucsa adót vetettek ki: ha az óra 12 fontnál kevesebbe került, az utána kifizetett adó a vételár egyharmada volt; ha az ára 12 és 16 font között volt, 4 font adót kellett fizetni; ha pedig 16 fontnál többe került, az adó a vételár egynegyede volt.

Írjon egy olyan programot, amely beolvas egy óra árlistát, melynek végét Ø-val jelezzük, és kiírja a fizetendő összeget (beleértve az óra árát és adóját).

Felhívjuk a figyelmét, hogy ebben a programban a cikluson belül lesz egy vagy több PRINT utasítás, és nincs szükség összegző blokkra. Egy jó folyamatábra feltétlenül szükséges lesz.

c) Írjon egy olyan programot, amely beolvas egy számsort, melynek végét Ø jelzi, és kiírja közülük *a legnagyobbat.*

Hasznos tanács: használjon egy változót, amelyben tárolja az addigi legnagyobb számot, s minden ciklusban vizsgálja meg, át kell-e értékét írni.

Vége a 14.2 Gyakorlatnak

Most ellenőrizze válaszait a B. Függelékben!

15. LECKE

Ez a lecke a Commodore BASIC három olyan fontos tulajdonságával foglalkozik, amelyeknek hasznát tudjuk venni a játékokban, kérdés–felelet, és más programokban, ahol a gép és használója szorosan együttműködik. Először is nézzük meg a "REAKCIO"-t, azt a programot, amely a könyvünkhöz tartozó szalagon vagy mágneslemezen található. A „reakcióidővel" lehet mérni, hogy az emberek milyen gyorsan reagálnak egy váratlan eseményre. Ahhoz, hogy valaki jó autóvezető legyen, és időben lenyomhassa a féket, ha egy gyerek átszalad az úton az autó előtt, gyors reakcióidőre van szüksége. A jó reakcióidő a legtöbb sportban és sok foglalkozásnál is igen hasznos tulajdonság.

Az emberek többségének 0,2 és 0,3 másodperc között van a reakcióideje, amikor összpontosít. A 0,2 másodpercnél kisebb reakcióidő nagyon gyorsan reagáló emberre vall, a 0,3 feletti pedig arra, hogy az illető egy-két pohárral többet ivott a kelleténél.

15.1 Gyakorlat

Töltse be a REAKCIO nevű programot, és segítségével mérje meg a saját reakcióidejét. Futtassa le többször a programot, és az első két-három eredményt ne vegye figyelembe, mert ez nem mérvadó. Próbálgassa a programot, amíg úgy érzi, hogy tökéletesen ért mindent; ekkor már teljes bizalommal felhasználhatja a programot arra, hogy lemérje vele egy olyan barátja reakcióidejét, akinek fogalma sincs a számítástechnikáról.
A programnak három olyan tulajdonsága van, amely első ránézésre nem nyilvánvaló:

Az első tulajdonság az, hogy amikor az utasítás azt mondja: „egy billentyűt", ezt úgy kell érteni, hogy bármelyiket. A funkció billentyűk, mint a [RETURN] és az [INST DEL] ugyanúgy működnek, mint a betűk és a számok, azonban a [C=], [SHIFT] és [CTRL] hatástalan. Csak a [f1 f4], [f2 f5], [f3 f6] és [HELP f7] feliratú hosszú billentyűket nem szabad használni, mert ha ezeket lenyomjuk, különböző karakterek jönnek létre.

A második tulajdonság az, hogy mindig különböző ideig kell várni, amíg a hangot meghalljuk; az idő egytől hat másodpercig tarthat, véletlenszerűen.

Végül a harmadik tulajdonság az, hogy ha a hang megszólalása előtt nyomunk le egy gombot, ezt az üzenetet kapjuk:

„Túl korán!"

Vizsgáljuk meg részletesen a program működését. Kezdjük az itt leírt folyamatábrával és a BASIC programmal.



1. felhő

2. felhő

3. felhő

4. felhő

5. felhő

6. felhő

7. felhő

1. felhő

```
10 REM REAKCIO IDO PROGRAM
20 PRINT " SHIFT és CTRL "
30 PRINT "REAKCIOIDEJENEK
   MERESEHEZ ÜSSÖN LE"
40 PRINT "EGY BILLENTYÜT ÉS
   VÁRJON A HANGRA"
50 PRINT "AMIKOR MEGHALLJA,
   ÜSSÖN LE EGY"
55 PRINT "BILLENTYÜT OLYAN
   HAMAR, AHOGY"
60 PRINT "CSAK TUDJA. JÓ
   SZERENCSÉT!"
```

2. felhő

```
70 REM VARAKOZAS
   BILLENTYURE
80 GET A$
90 IF A$ = "" THEN 80
```

3. felhő (és lásd még alább)

```
100 REM VELETLEN HOSSZU
    VARAKOZAS
110 PRINT
120 PRINT "VÁRAKOZZON
    A HANGRA"
130 PRINT
140 Q=TI+INT(60+301*RND(0))
150 GET A$
160 IF A$ <>"" THEN 310
170 IF TI < Q THEN 150
```

4. felhő

```
180 REM HANG INDUL, IDO
    FELJEGYEZESE
190 VOL 7
200 SOUND 1, 950 4
210 X=TI
```

5. felhő
```
220 REM BILLENTYURE
    VARAKOZAS
230 GET A$
240 IF A$ = "" THEN 230
```
6. felhő
```
250 REM EREDMENY ROGZITESE
260 R=TI
```
7. felhő
```
270 REM EREDMENY KIJELZESE
280 PRINT "REAKCIOIDEJE:"
290 PRINT (R – X)/60;
    "MÁSODPERC"
300 STOP
```
A 3. felhő része
```
310 PRINT "TUL KORÁN"
320 STOP
READY
```

A programhoz olyan jelzéseket tettünk, hogy a folyamatábra felhőinek megfelelő utasításokat világosan átláthassuk.

Az első felhő (10...60 sor) kizárólag PRINT utasításokból áll, és eléggé nyilvánvaló.

A második felhő (70...90 sor) addig várakoztatja a programot, amíg a használó leüt egy billentyűt. Ebben a felhőben az egyik utasításnak eddig ismeretlen kulcsszava van:

GET A$

Ez az utasítás bizonyos mértékig hasonlít az INPUT-ra; továbbítja az információt a billentyűzetről a számítógéphez. A kettő között azonban lényeges különbségek is vannak:

1. A GET kulcsszó után egyetlen változó név állhat. Például:

GET Q
GET X$
GET PR$
(megengedett)

de

GET Y$, Z$ tilos

Nem BASIC nyelven van!

2. A GET utasítás nem vár arra, hogy a használó valamit csináljon; egyszerűen megvizsgálja a billentyűzetet abban a pillanatban, és azt jelzi, hogy melyik billentyű van éppen lenyomva. Ha van lenyomott billentyű, a megfelelő karakter vagy számérték bekerül a GET utasításban szereplő változóba. Ha éppen nincs lenyomott billentyű, a változó értéke az üres füzér vagy null karakter (nem tévesztendő össze a nulla vagy zérus számmal) lesz, ha a GET utasítás változója füzérváltozó, és Ø, ha a változó numerikus. Az üres füzérben nincsenek karakterek [Pontosabban a null karakter van benne *(a lektor megjegyzése)*.], jelölése: "" Megjegyzés: ebből következik, hogy nem célszerű a GET használata numerikus változóval, mert nem tudja megkülönböztetni a Ø billentyűt a lenyomás nélküli állapottól.

Fentiek szemléltetésére képzeljük el, hogy a gépen elindítjuk a következő ciklust tartalmazó programot, és figyeljük meg, mi zajlik le a gép belsejében:

```
10 GET X$
20 GOTO 10
```

A számítógép körülbelül 50-szer megy át ezen a cikluson egy másodperc alatt. Amíg a használó nem üt le valamit a billentyűzeten, az X$ értéke az üres füzér:"". Tegyük fel, hogy a használó most leüt egy billentyűt, mondjuk az U betűt. Amint a gép a GET utasítást végrehajtja (a másodperc ötvened része alatt), az X$ értéke az "U" füzér lesz. Ez azonban csak egyszer történik meg minden billentyű leütésekor; a következő ciklusnál az X$ értéke újra a "" füzér lesz. Ez addig tart, amíg az U billentyűt el nem engedik, és egy másik, esetleg ugyanazt a billentyűt le nem nyomnak; kivéve, ha túl sokáig tartjuk a billentyűt lenyomva, és emiatt ismétli magát. [Próbáljuk ki: ha egy billentyűt kb. 1 mp-nél tovább tartunk lenyomva, elkezdi gyorsan ismételni magát *(a lektor megjegyzése)*.] Ekkor ugyanaz játszódik le, mintha elengedtük volna, és újra lenyomtuk volna.

3. A GET utasítás nem kezeli speciális esetként a vezérlő karaktereket, mint például az INST DEL -t, a RETURN -t vagy a kurzor vezérlőket, kivéve a RUN STOP -ot, amely megszakítja a programot.

4. A GET utasítás által észlelt bármelyik karakter nem jelenik meg a képernyőn.

Ezeknek az ismeretében most már érthető a REAKCIÓ program 8Ø és 9Ø sora. A 8Ø-as utasítás megvizsgálja a billentyűzetet és az A$-ba mindaddig üres füzért tárol, míg le nem nyomnak egy billentyűt. A 9Ø utasítás ellenőrzi az A$-t, és visszaküldi a gépet a 8Ø-ra mindaddig, amíg a használó be nem gépel valamit; ekkor pedig megengedi a programnak, hogy továbblépjen a 1ØØ. sorra.

Ennek a felhőnek az a feladata, hogy feltartsa a programot mindaddig, amíg a használó jelzi, hogy felkészült a reakcióidejének mérésére. Miért használunk egy ciklust a GET utasítással egy ilyen egyszerű utasítás helyett:

```
INPUT "READY"; A$
```

Ennek két oka van. Az első az, hogy az INPUT mindig vár egy RETURN utasítást, miután megkapta a használótól az üzenetet. Ehhez legalább két karaktert kell beütni.

A másik az, hogy a GET utasítás majdnem mindegyik karaktert egyformán kezeli, ezért sokkal kisebb a valószínűsége, hogy a program rosszul működik, ha a használó egy vezérlő billentyűt üt le szám vagy betű helyett.

A 3. felhő arra készteti a gépet, hogy várja meg azt a meghatározatlan véletlenszerű időt, ami a felhasználó "kész" jelzése és a gép hangjelzése között eltelik. A várakozási időnek mindig másnak kell lennie, mert ha mindig ugyanannyi lenne, a használó hamar megtanulná, hogy mennyit kell várnia a hangjelzés megszólalásáig, és akkor már nem volna benne semmi váratlan. Ez a felhő két olyan adottsággal rendelkezik, amelyekkel még nem találkoztunk: a *véletlen* funkció és a *belső óra*.

A véletlen függvénnyel tudunk a gépből kiváltani egy *előre ki nem számítható* számot. (A szám valójában nem előre kiszámíthatatlan, mert ami a számítógépben történik, az mindig attól függ, hogy mi történt előtte. Azonban minden új véletlen szám az előzőből négyzetreemelés és számjegylevágás műveletek bonyolult folyamatával kerül előállításra, és ezt a folyamatot nem ismerjük pontosan, nem tudjuk megmondani, hogy milyen szám lesz a következő.) Az RND (Ø) (RaNDom, azaz véletlen rövidítése) hatására a számítógép minden esetben egy különböző, 0 és 1 közé eső számértéket állít elő.

A gyakorlati esetek többségében nem egy Ø és 1 közötti véletlen *tört* számára van szükségünk, hanem olyan *egész számra,* amelynek határait a megoldandó probléma jellege határozza meg. Ha például egy 6 oldalú kockával játszó embert akarunk a géppel utánoztatni, 1 és 6 közötti számot várunk; a rulett kerekéhez pedig egy Ø és 36 közötti számra van szükség.

Ahhoz, hogy egy meghatározott értékhatáron belüli egész számot kapjunk, egy kicsit eltérő kifejezést használunk:

```
INT (x+y*RND(Ø))
```

ahol x a *legkisebb* szám, amire szükségünk van,
y *a különböző lehetőségek száma.*

Tehát ahhoz, hogy egy 1 és 6 közötti számot kapjunk, a következőt kell beírnunk:

A várt legkisebb szám

```
INT (1+6*RND(Ø))
```

vagy egy Ø és 36 közötti számhoz

```
INT (Ø+37*RND(Ø))
```

A különböző értékek száma

Egy ilyen jellegű kifejezést ciklusba is írhatunk, és akkor a gép többször állítja elő. Írja be a következő programot, amely 120 kockadobást szimulál:

```
NEW
1Ø FOR J=1 TO 12Ø
2Ø S=INT (1+6*RND(Ø))
3Ø PRINT S;
4Ø NEXT J
5Ø STOP
```

Futtassa le ezt a programot, és számolja meg a képernyőn megjelenő 1-esek, 2-esek...6-osok számát. Írja be az eredményeit az alábbi táblázat első sorába:

	1	2	3	4	5	6
Dobások száma (1)						
Dobások száma (2)						

Jól utánozza a program egy rendes (nem hamisított) kocka dobásait?

Futtassa le másodszor is a programot, és töltse ki a második sort. Figyelje meg az eredményeket; észre fogja venni, hogy különböznek az első sorozattól, mint az az igazi kockadobásnál is várható.

A 3. felhő másik érdekes tulajdonsága a belső óra, a TI (TIme = idő). A TI$ órával, amely az időt órákban, percekben és másodpercekben méri, már találkoztunk; de a TI speciális változónak (amely nem egy füzér, hanem szám) az a feladata, hogy sokkal rövidebb időegységeket mérjen. Amikor a számítógépet bekapcsoljuk, a TI nullára van állítva, és attól kezdve a gép 1-et hozzáad a másodperc minden 60-ad részében, függetlenül attól, hogy mi zajlik le még ezen kívül a gépben. A belső óra pillanatnyi értékét bármikor megkaphatjuk hatvanadmásodpercben úgy, hogy egy kifejezésben használjuk a TI-t; de ezt az értéket nem lehet úgy megváltoztatni, mint ahogy a TI$ értéket beállíthatjuk. Gépeljük be a

```
PRINT TI
```

parancsot. A gép úgy válaszol, hogy kiír egy meglehetősen nagy számot (6Ø*6Ø, azaz 36ØØ hatvanadmásodpercet minden percre, amióta a gép be van kapcsolva). Ismételjük meg a parancsot, és figyeljük meg, hogy az érték egy-néhány százzal megnövekedett. Végezetül próbáljuk nullázni (reset) a TI értékét, és nézzük meg, mi történik!

A TI-t arra használhatjuk, hogy az időtartamot mérjük két különböző, de egymással összefüggő módon. Egyik esetben sem az az érdekes számunkra, hogy hány hatvanadmásodperc telt el a számítógép bekapcsolása óta; itt azt a tényt használjuk fel, hogy bármilyen hosszúságú időtartamot megadja a végén mért TI és az indulása pillanatában mért TI értékek közötti különbség. Például egy 5 másodperces időtartam végén a TI értéke 5*6Ø-nal, azaz 3ØØ-zal lesz több, mint induláskor volt. Ez attól függetlenül igaz, hogy a gép 5 másodperce vagy 5 órája van bekapcsolva.

A belső óra használatának egyik módja, amikor ismert időtartamot méretünk meg a géppel, és a gépnek jeleznie kell, hogy mikor telt le ez az idő. A módszer egyszerű. Az időtartam elején a program megnézi a TI-t, és kiszámítja, hogy mennyinek kell lennie az időtartam végén; aztán egy ciklussal addig vár, amíg a TI eléri (vagy túlhaladja) azt az értéket. [A pontos elérés nem használható vizsgálatként. Míg a ciklusban végigmegy a gép, több hatvanadmásodperc eltelhet, s vizsgálatkor nem biztosítja semmi, hogy éppen a várt érték lesz *(a lektor megjegyzése)*.] A helyzet nagyon hasonlít ahhoz, amikor a konyhában is számolunk: "Ennek a krumplinak 25 percig kell főnie. Most 10 perccel múlt 4 óra, úgyhogy 4 óra 35 perckor leveszem majd a tűzről."

A dolog szemléltetésére íme egy általános időmérő program, amit egyformán használhatunk pl. a konyhában, a laboratóriumban stb.:

```
1Ø INPUT "HANY PERC"; P
2Ø R=TI+P*36ØØ
3Ø IF TI<R THEN 3Ø
4Ø PRINT "IDO LEJART!"
5Ø STOP
```

Amikor kipróbálja a programot, kevés számú perccel dolgozzon, hogy ne kelljen nagyon sokáig várnia. A program tanulmányozása közben ne felejtse el, hogy a TI értéke állandóan növekszik, és végül a P*36ØØ hatvanadmásodperc után a TI<P feltétel *hamis* lesz.

A másik esetben azt várjuk a számítógéptől, hogy mondja meg, *mennyi idő* telik el egy adott pillanattól addig, amíg egy másik esemény bekövetkezik. Tároljuk egy változóban a TI értékét a mérendő időszakasz elején. Amikor a másik esemény bekövetkezik, a TI akkori és rögzített érték közötti különbség az időtartam hatvanadmásodpercekben megadott értéke. Olyan ez, mint mikor a hegymászó azt mondja: "Emlékszem, hogy 5 órakor kezdtem megmászni azt a hegyet. Tizenegykor értem a csúcsra, tehát hat óra alatt másztam meg."

Az olyan programnak, amely ezzel a módszerrel méri az időt, efféle utasításokat kell tartalmaznia:

R=TI (Az időtartam kezdeténél mért TI értéket tárolja)

majd később

E=TI (Megszerzi a TI értékét az időtartam végén)

D=E−R (Megállapítja a különbözetet hatvanadmásodpercben)

S=D/6Ø (Megállapítja az időkülönbséget másodpercben)

PRINT "EZ";S; "MÁSODPERCBE TELLETT"

Most már összerakhatjuk a 3. felhő utasításainak darabjait.

A várakozási időt 1 és 6 másodperc között akarjuk megállapítani. Ez 6Ø és 36Ø hatvanadmásodperc között lesz, amelyet véletlenszerűen *a gép* fog meghatározni. A megfelelő kifejezés:


```
INT (6Ø+3Ø1*RND(Ø))
```

A várakozási időt közvetlenül az időtartam kezdete előtt határozza meg a gép, tehát előre *ismert* (bár a használó nem tudja). Az elsőként ismeretetett időmérő módszert használjuk, amely előre megmondja, hogy menyi lesz a TI értéke az időszakasz végén. Ezt a 14Ø. utasítás végzi el, és Q-ban rögzíti az értékeket.

Ha csak ennyire volna szükségünk, ilyen lehetne a teljes felhő:

```
14Ø Q = TI+INT(6Ø+3Ø1*RND(Ø))

17Ø IF TI<Q THEN 17Ø
```

Azonban ellenőriznünk kell, hogy a használó nem ütött-e le valamilyen billentyűt, mielőtt a hangjelzés megszólalt volna. A 15Ø, 16Ø, 31Ø és 32Ø utasítások egyszerűen azért vannak a programban, hogy ezt a lehetőséget ellenőrizzék.

A program további része teljesen nyilvánvaló. Az X tárolja az időszakasz kezdetén meglévő TI értéket, a 23Ø és 24Ø utasítás pedig arra vár, hogy a használó leüssön egy billentyűt.

Tanulmányozza át alaposan a programot, amíg minden utasítást biztosan meg nem ért.

Vége a 15.1 Gyakorlatnak

15.2 Gyakorlat

1. Írjon egy stopperóra programot. A program akkor kezdi el az időmérést, amikor a használó lenyomja az I billentyűt. Amikor az S-t benyomja, a program leáll, és másodpercekben kijelzi a letelt időt. A programnak ki kell írnia a használati utasítást, hogy minden további magyarázat nélkül bárki használhassa. Hasznos tanács: alkalmazza a GET-et és a TI-t.

2. Írjon egy olyan programot, amely egy pénzfeldobós játékot utánoz. Amikor a játszó személy leüt egy gombot, a program véletlenszerűen vagy „FEJ"-et, vagy „ÍRÁS"-t jelez ki.

Vége a 15.2 Gyakorlatnak

15.3 Gyakorlat

A véletlenszámokat jól fel lehet használni az olyan szerencsejátékokban, mint a kockajáték, a félkarú rabló stb. Minden ilyen program azonos mintázatra épül, amely egy dobás-ra vagy pörgetés-re a következő:

Kérdezze meg a használótól tétje részleteit

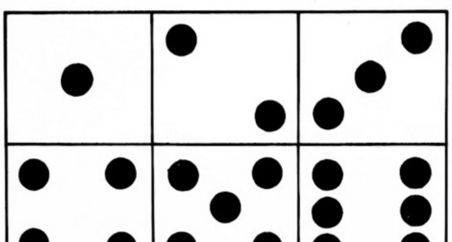
↓

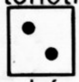
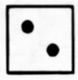
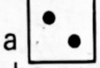
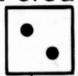
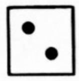
Használjon véletlenszámokat a dobások, pörgetések stb. utánzására

↓

Döntse el, hogy a játékos nyert-e, és közölje vele az eredményt

Szemléltessük az elvet a régi „korona és horgony" játékkal, amelyet három kockával játszanak egy olyan táblán, amely 6 négyzetre van osztva. (A korona és horgony kocka általában más jelekkel van ellátva, de ez a játék elvét nem befolyásolja.)



A játékos a tétjét bármelyik négyzetre teheti. Feltehet például 5 fontot a kettes  négyzetre. Ekkor a bankos dob mind a három kockával. Ha egy kocka mutat -t, a játékos kétszeresen kapja vissza a tétet. ha két kockán jön ki a , a játékos az eredeti tét háromszorosát, ha pedig a  mind a három kockán kijön, a játékos jutalma a tét négyszerese. Minden jutalom tartalmazza az eredeti tétet. Ha viszont egyetlen  sem jön ki, a játékos elveszti a tétet.

Közlünk a korona és horgony játék egy dobásához készített programot; a változójegyzék használatával gond nélkül tudja a program működését követni.

S: A játékos tétje
N: A játékos által választott szám (amire tett)
D1, D2, D3: A 3 kocka dobásainak eredménye
C: Azoknak a kockáknak a száma, amelyeken kijön az N, a fogadó által választott szám

```
10 INPUT "TETJE"; S

20 INPUT "A SZAM AMIRE TESZI
                        (1–6)";N      ⎫
30 D1 = INT (1+6*RND(Ø))              ⎬ A 3 kocka dobásai
40 D2 = INT (1+6*RND(Ø))              ⎪
50 D3 = INT (1+6*RND(Ø))              ⎭

60 C = Ø                              ⎫
70 IF D1<>N THEN 9Ø                   ⎪
80 C = C+1                            ⎪
90 IF D2<>N THEN 11Ø                  ⎬ Megszámolja a fo-
100 C = C+1                           ⎪ gadott számokat a
110 IF D3<>N THEN 13Ø                 ⎪ kockákon
120 C = C+1                           ⎭
```

Kijelzi az eredményt

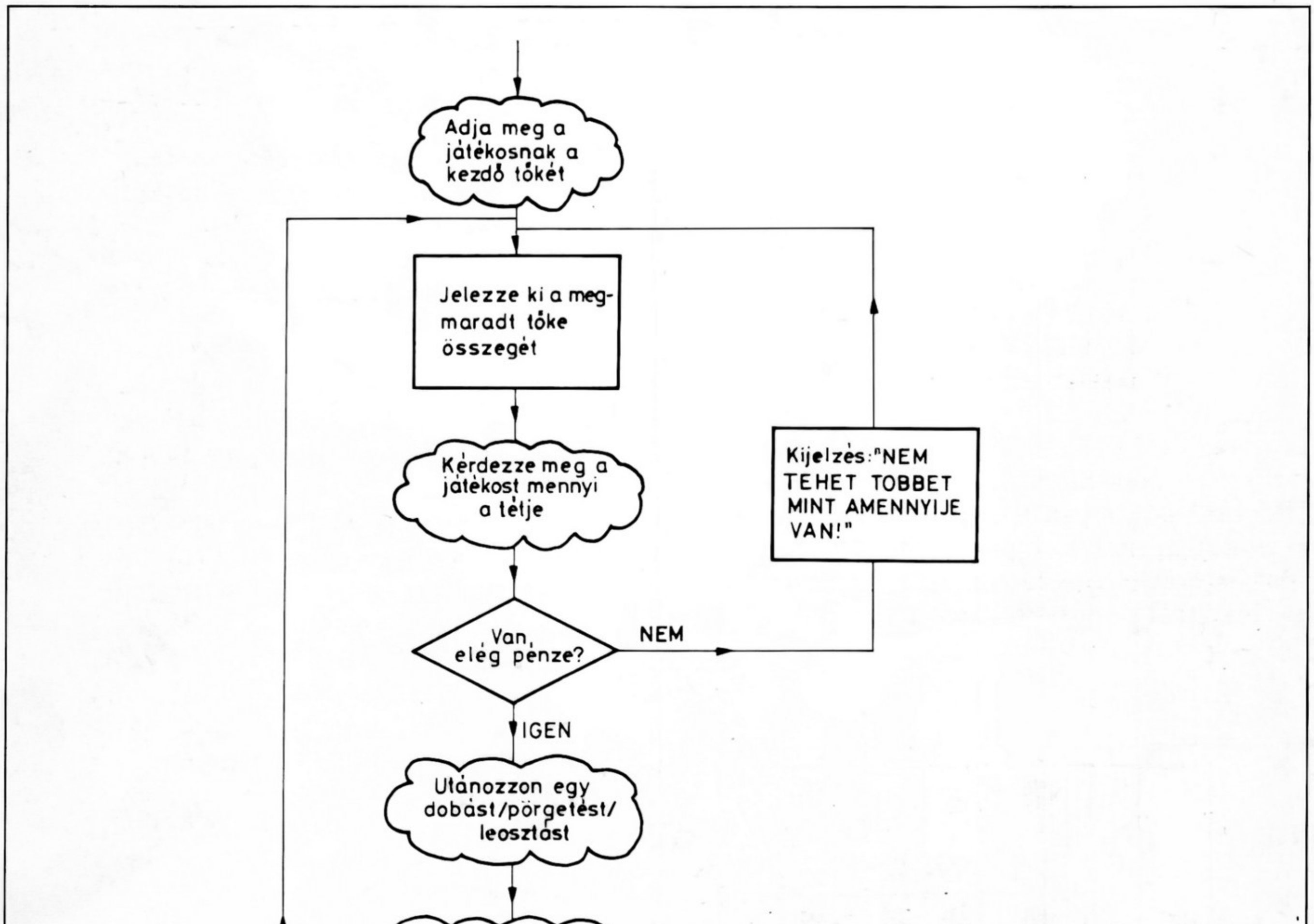
```
130 PRINT "KOCKAK DOBASAI:";
D1; D2; D3
140 IF C<>0 THEN 170
150 PRINT "VESZTETT"
160 GOTO 180
170 PRINT "KAP";S (C+1); "FONTOT"
180 STOP
```

Ritka az olyan játékos, aki az első dobás után azonnal abbahagyja a játékot. Az emberek általában egy bizonyos összeggel kezdenek el játszani, és addig folytatják, amíg az összes pénzt el nem vesztik, vagy – nagyon ritka esetben – amíg a bank tönkre nem megy.

A szerencsejáték-programok akkor jók, ha ilyen teljes játékmeneteket utánoznak. Kezdésnek a játékos kap egy bizonyos 'pénz'-összeget (mondjuk 100 fontot), és ezzel addig játszhat, amíg kedve tartja, vagy amíg el nem fogy minden pénze. Megadjuk egy ilyen játék folyamatábráját:

A folyamatábra segítségével alakítsa át a korona és horgony programot így, hogy a játékos 100 font tőkével kezdjen, és addig játszhasson, ameddig akar. Amikor a program készen van, futtassa le többször és döntse el, hogy mi lenne szívesebben: játékos vagy bankár.

Vége a 15.3 Gyakorlatnak



15.4. Gyakorlat

Írjon egy olyan programot, amely bármilyen Ön által ismert szerencsejátékot utánoz. Színesítse a programot kockák vagy kártyák képével, hozzájuk illő hangokkal stb.

Vége a 15.4 Gyakorlatnak

A könyvhöz tartozó kazetta, illetve mágneslemez a kockajáték egyik változatát tartalmazza; próbálja ki! A KOCKAJÁTÉK nevű program teljes listáját a B. Függelékben találja meg.

UTÓSZÓ

Ezzel Ön elérkezett a tanfolyam végére – gratulálok! Ezzel a programozás alapelveinek egy jó ismeretéhez jutott, és olyan programokat képes tervezni és írni, amelyek az érdekes problémáknak és a számítógép alkalmazásának széles területét ölelik fel. Remélem, hogy az alapos, megfontolt tervezés, a folyamatábrák, változójegyzékek és a programokra vonatkozó megjegyzések megőrzése és rendszerezése is begyakorlott szokássá vált.

A tervezésnek és a saját munkája megszervezésének a minősége az, amely az igazán hozzáértő programozót megkülönbözteti a többitől.

Ezen a ponton Ön félúthoz érkezett a BASIC nyelv tanulmányozásában. Sok olyan fontos probléma van, amelyek megoldásához a nyelv olyan részeire van szüksége, amelyeket Ön még nem ismer. Ilyen például egy mozgófilm készítése a képernyőn, vagy nevek abc szerinti rendezése vagy ezek tárolása mágnesszalagon vagy mágneslemezen. Ezeket a témákat tárgyalja alaposan – sok egyéb között – ennek a sorozatnak a második kötete, melynek címe: BEVEZETÉS A BASIC NYELVBE (II. RÉSZ). Ez a könyv is abban a stílusban íródott, mint amilyenben az, amelyet most befejezett, és a BASIC nyelv ismeretét teljesre egészíti ki.

Mint ahogy a bevezetőben is mondtuk, a programozás nagyon széles körű tárgy. Miután a kezdeti lépéseket megtette, 3 módon mélyítheti tovább tudását:

a) Olvasson el mindent, ami a tárgyra vonatkozik. Számos folyóiratot találhat, ahol programokat közölnek, magyaráznak. A programozásról szóló könyveket akkor is érdemes elolvasni, ha nem utalnak speciálisan egy-egy számítógépre.

b) Lépjen be egy számítógépes körbe, szakkörbe vagy μ-klubba. Művelődési házakban, közösségi házakban, iskolákban található ilyeneket.

c) Foglalkozzon a programkészítéssel, gyakoroljon állandóan, és törekedjen hibátlan munkára. Úgy tervezze meg a programokat, hogy azok megbízhatóak legyenek, és bárki fel tudja használni őket külön útmutatás nélkül.

Úgy írja meg a programokat, hogy ne szégyenkezve, hanem büszkén mutathassa meg egy másik számítógépes szakembernek.

És még valami. Ön most egy ragyogó hobbit, de talán egy életre szóló szakmát is talált magának. Ne felejtse el, hogy a számítógépek ismerete nemcsak előnyt jelent, de annak a felelősségét is magával vonja, hogy a gépet humánusan és okosan használja fel. Senki sem vágyik egy olyan számítógép irányította társadalomra, ahol keveset kell dolgozni és az emberek nem szabadok, többek közt *Önön* is múlik, hogy ettől megmeneküljünk.

FÜGGELÉK

A. Függelék

Számítógépe nagyméretű matematikai feladatok elvégzésére alkalmas; történeti érdekességként elmondjuk, hogy jóval gyorsabban végzi el az aritmetikai feladatokat, mint az 1960 előtt felszerelt nagyméretű számítógépek többsége!

Ebben a függelékben vázoljuk a számítógép néhány matematikai jellegzetességét. Ha matematikai, természettudományos vagy gépészeti számításokhoz akarja használni a számítógépet, csak el kell olvasnia és megértenie a függelék anyagát. Néhány itt leírt tulajdonsága egyszerű, és könnyen felfoghatja bárki, aki még emlékszik az iskolában tanult elemi aritmetikára. Néhány dologhoz viszont alaposabb matematikai tudásra van szükség. Csak addig haladjon, ameddig tudása és magabiztossága viszi, de azt feltételezzük, hogy a könyv minden leckéjét végigolvasta.

1. KIFEJEZÉSEK

Azok a kifejezések, amelyeket a 4. leckében említettünk először, nagyon egyszerű példái egy sokkal általánosabb lehetőségnek. Ezekben a parancsokban például:

A = <u>34</u>

B = <u>B+1</u>

C = <u>((X+Y)–34.7/(Q–3))∗(Z–3)↑2</u>

Az aláhúzott részek mind olyan kifejezések, amelyeket a számítógép feldolgoz a kedvünkért.

A kifejezések három típusú elemből épülnek fel:

Értékek: numerikus változók vagy számok mint

B, X, Y, 34, 34.7

Műveleti jelek: a + – ∗ / és ↑ jelek

(↑ jelentése: hatványozás)

Zárójelek: (és)

A BASIC-ben ugyanúgy írjuk a kifejezéseket, mint a rendes algebrában, és ugyanazt jelentik. Négy apróbb különbség van:

– A BASIC kifejezéseket nagy betűvel írjuk, és nem kicsivel.*

– A hatványozást a ↑ jellel kell kifejezni, mert a számítógéppel nem lehet kis számokat írni a vonal fölé. A 3^2 helyett így írjuk: 3↑2.

– A szorzást mindig a ∗ jellel kell kifejezni. A BASIC-ben azt írjuk, hogy "3∗A", és nem "3A", mint a hagyományos algebrában. Ennek a szabálynak a megsértése olyan hibákat okozhat, amelyeket nehéz észrevenni. Ha BA-t írunk, akkor B∗A-ra gondolunk, a gép azt feltételezi, hogy egy új BA *nevű* változóról van szó. Szintaktikus hibát nem jelez, de rossz választ ad!

– Az osztás A/B és nem A:B vagy $\frac{A}{B}$. Ha az osztás számlálója vagy nevezője összetett kifejezés, zárójellel kell őket elhatárolni. A $\frac{3+5}{7+8}$ helyes leírása a BASIC-ben (3+5)/(7+8). Ha kihagyjuk a zárójelet, és így írjuk le: 3+5/7+8, a műveletek sorrendjére vonatkozó törvény szerint (amelyről a következő bekezdésben beszélünk) a gép ilyen kifejezésként kezeli:

$$3+\frac{5}{7}+8.$$

Amikor a számítógép elvégez egy műveletet, először a ↑ jelet veszi sorra, aztán a szorzásokat és osztásokat, végül pedig az összeadásokat és a kivonásokat; minden

* Hacsak nem kisbetű/nagybetű módban vagyunk, mint pl. a magyar ékezetes betűk alkalmazásakor. Ekkor mindig kis betűvel írjuk (a lektor megjegyzése).

esetben balról jobbra haladva. Mindig azt számolja ki elsőnek, ami zárójelen belül van. Ezeket hívjuk a műveleti sorrendiség (precedencia) szabályának, és ezek *ugyanazt* az eredményt adják, mint a rendes iskolai algebra.

A kifejezésekben szereplő számok értékének nem kell feltétlenül egész számnak lennie, lehetnek tizedesszámok is. A számítógép körülbelül 8 tizedes pontossággal dolgozik, ami azt jelenti, hogy sok törtet (mint pl. 1/3 vagy 1/7) nem lehet teljes pontossággal ábrázolni. Néhány aritmetikai utasításban számíthatunk kisebb „kerekítési" hibákra, tehát lehet, hogy egy pontosan 7-nek várt eredmény így jön elő: „6.99999998".

Annak ellenőrzésére, hogy jól érti a kifejezéseket, számolja ki a következő példákat, és mondja meg előre, hogy a számítógép mit fog kijelezni az egyes esetekben. Tételezze fel, hogy X = 3, P = 7.

Parancs	Megjósolt eredmény	Tényleges eredmény
PRINT 3+12−6−4		
PRINT 4+3*2		
PRINT X+P−3		
PRINT 5+12/6−3		
PRINT 11/5−7/4		
PRINT 4↑2−2↑4		
PRINT 3+2↑3−3↑2		
PRINT 2↑X−P		
PRINT 3+12−(6−4)		
PRINT 5+12/(6−3)		
PRINT (P+X)↑(1−X)		
PRINT 4↑2−3↑Ø		
PRINT (P↑2−X↑2)/3		

Ezután ellenőrizze az eredményeket a számítógépen. Ne felejtsen el az indulás előtt a P és X változóknak értéket adni.

A BASIC-ben a kifejezéseket leggyakrabban a PRINT és a LET utasításokban használják. Íme egy egyszerű program, amely beolvas két U és V számot, és kijelzi az F „lencse" képlettel kiszámított értékét:

$$\frac{1}{F} = \frac{1}{V} + \frac{1}{U}, \text{ vagy } \frac{1}{\frac{1}{V} + \frac{1}{U}}$$

```
10 INPUT "V";V
20 INPUT "U";U
30 PRINT "F="; 1/(1/V+1/U)
40 STOP
```

1. Példa

Írjon egy olyan programot, amely beolvas két értéket, V-t és R-t, és kiírja a $A = \frac{V^2}{R}$ képlet értékét.

2. Példa

Írjon egy olyan programot, amely kiszámítja az $y = \frac{1}{1+x^2}$ képlet értékét az x, Ø és 2 közötti értékeire, 0.2-es lépésközzel.
(Hasznos tanács: használjon ehhez hasonló FOR ciklust:

```
FOR X=Ø TO 2 STEP Ø.2

...

NEXT X)
```

(A helyes válaszok a B. Függelék végén találhatók.)

2. GYAKORI FÜGGVÉNYEK

A számítógépnek is van egy sor „tudományos" függvénye, csakúgy, mint a legtöbb kalkulátornak. Az egyik hasznos függvény a négyzetgyök. A SQuare Root (= négyzetgyök) rövidítése SQR, és pl. ilyen kifejezésekben állhat:

```
PRINT SQR(5)

PRINT SQR (B↑2+C↑2)
```

A zárójelben levő mennyiséget a függvény argumentumának hívják. A SQR esetében ennek nullának vagy pozitív számnak kell lennie.

Az alábbi program kiírja minden 100 és 115 közötti szám négyzetgyökét.

```
1Ø PRINT "N"; "SQR(N)"

2Ø FOR N = 1ØØ TO 115

3Ø PRINT N; SQR(N)

4Ø NEXT N

5Ø STOP
```

3. Példa

Ha egy háromszög három oldalának hossza a, b és c, akkor a háromszög területét ez a képlet adja meg: $a = \sqrt{s(s-a)(s-b)(s-c)}$, ahol s a háromszög kerületének fele, $(a+b+c)/2$.

Írjon egy olyan programot, amely beolvas három számot. Ha ezek egy valódi háromszög oldalai lehetnek, a program kiírja a háromszög területét, ha nem (tehát ha a számok pl. ilyenek: 1,1, 1Ø), a program megfelelő üzenetet ír ki.

(Hasznos tanács: ha a szakaszok nem alkotnak háromszöget, az $s(s-a)(s-b)(s-c)$ értéke negatív!)

Most ellenőrizheti válasza helyességét a B. Függelékben!

A következőkben megadunk néhányat a legfontosabb matematikai függvények közül. Olvassa végig őket, de ne tartsa kötelességének, hogy mindet megtanulja kívülről. Amikor szüksége lesz rájuk, visszalapozhat ide.

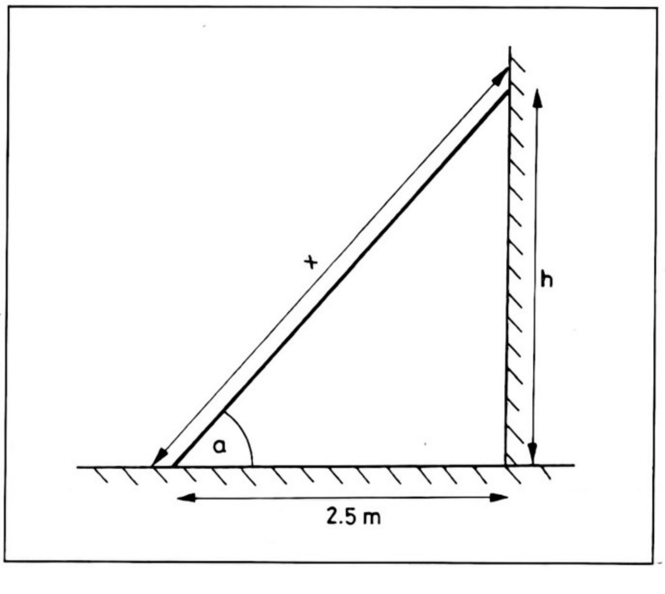
SIN(X)	Trigonometriai függvények
COS(X)	Az argumentumot *radiánban* kell megadni
TAN(X)	(1 fok = π/180 radián)
ATN(X)	X arkusz tangense. Az eredményt radiánban kapjuk, a −π/2 és π/2 intervallumban.
LOG(X)	X természetes logaritmusa (e alapú logaritmus), X-nek pozitívnak kell lennie.
EXP(X)	Ez az e^x függvény
ABS(X)	X abszolút értéke (X, ha X≧0; egyébként −X).
INT(X)	A legnagyobb egész szám, amely kisebb vagy egyenlő mint X. Figyeljen arra, hogy INT(3.5)=3 INT(−3.5)=−4

A 3.14159265... szám helyett a billentyűzet π jelét is használhatja.

Íme egy példa annak bemutatására, hogy milyen egyéb felhasználási lehetőségei vannak e függvényeknek.

Egy létra magasságát 4 és 5 méter között lehet változtatni, 20 centiméterenként. Az alja 2,5 méterre van egy függőleges faltól, a teteje pedig a falnak van támasztva. Írjon egy olyan programot, amely mind a 6 lehetséges létrahossz esetére megadja a létra vízszintessel bezárt dőlésszögét.

Először a matematikai részt végezzük el egy ábra segítségével. Az x jelöli a létra hosszát, a h a létra tetejének magasságát, a pedig a vízszintessel bezárt szöget.



$h = \sqrt{x^2 - (2,5)^2}$ (Pythagorasz tétele alapján)

$a = \arctan(h/2,5)$ (radiánban)

vagy

$a = (180/\pi) * \arctan(h/2,5)$ (fokban)

A következő lépésben megírjuk a programot, amelynek egyszerű ciklus szerkezete van:

```
10 PRINT " HOSSZ"," SZOG"
20 FOR X=4 TO 5 STEP 0.2
30 H=SQR(X↑2−2.5↑2)
40 A = (180/π * ATN(H/2.5)
50 PRINT X,A
60 NEXT X
70 STOP
```

Az egyik leghasznosabb függvény az INT. Arra használhatjuk, hogy megtudjuk, egy szám maradék nélkül oszt-e egy másikat. Ha X pontos többszöröse Y-nak, akkor a

X/Y = INT(X/Y)

feltétel igaz, egyébként nem.

Azt a számot, amelyet csak önmagával és 1-gyel lehet osztani, *prímszámnak* nevezzük. A következő program kiszámítja és kiírja a prímszámokat 3-tól felfelé addig az értékig, amelyet a használó megad:

```
10 INPUT "LEGMAGASABB ERTEK";L
20 FOR N=3 TO L
30 FOR J=2 TO N−1
40 IF N/J=INT(N/J) THEN 70
50 NEXT J
60 PRINT N;
70 NEXT N
80 STOP
```

4. Példa

Tanulmányozza alaposan a prímszám programot (ha szükség van rá, nyomkövetéssel), és állapítsa meg a működési elvét. Futtassa le és mérje le futási idejét L valamilyen értékére (mondjuk 500-ra).

Ezzel a módszerrel nagyon lassan lehet csak kiszámítani a prímszámokat. Tervezzen meg és építsen be a programba olyan javításokat, amelyektől gyorsabban fut a program.

Tanácsok:

a) A 2 kivételével egyetlen páros szám sem lehet prímszám.

b) A lehetséges tényezők ellenőrzésekor elég a szám négyzetgyökéig elmenni.

Ellenőrizze a válaszát a B. Függelékben!

B. Függelék

7. LECKE

7.1. GYAKORLAT

a) I, I, I, I, H, H, H
b) H, H, I, I

7.2 GYAKORLAT

Ciklus-változó	Kezdő-érték	Végérték	Növek-mény	Lefutott ciklusok száma
X$	"A"	"ABBB"	"B"	4
P	Ø	1Ø	+1	11
Y$	"Z"	"ZXYXY"	"XY"	3
R	5	14	3	4
C	27	7	−5	5

7.3 GYAKORLAT

1.
```
1Ø P$="*"
2Ø PRINT P$
3Ø P$=P$+"*"
4Ø IF P$<>"***********"
   THEN 2Ø
5Ø STOP
```

2.
```
1Ø PRINT "FONT", "DOLLAR"
2Ø PRINT
3Ø P=1Ø
4Ø PRINT P, 1.77*P
5Ø P=P+2
6Ø IF P<32 THEN 4Ø
7Ø STOP
```

3.
```
1Ø PRINT "CELS", "FAHR"
2Ø PRINT
3Ø C=15
4Ø F=1.8*C+32
5Ø PRINT C,F
6Ø C=C+1
7Ø IF C<31 THEN 4Ø
8Ø STOP
```

8. LECKE

8.1 GYAKORLAT

a) Ciklusszámláló ~~1Ø~~ ~~2Ø~~ ~~3Ø~~ ~~4Ø~~ ~~5Ø~~ 6Ø
Változók: X:5 Y:7 Z:12 W:2

```
5 7 12 2       1Ø X=5
BREAK IN 6Ø    2Ø Y=7
READY          3Ø Z=X+Y
               4Ø W=Y-X
               5Ø PRINT X;Y;Z;W
               6Ø STOP
```

Ciklusszámláló ~~1~~ ~~2~~ 3
Változók Q: ~~1~~ ~~2~~ 3

```
SZERET         1Ø Q=1
NEM SZERET     2Ø PRINT "SZERET"
NEM SZERET     3Ø PRINT "NEM SZERET"
BREAK IN 6Ø    4Ø Q=Q+1
READY          5Ø IF Q<3 THEN 3Ø
               6Ø STOP
```

8.2 GYAKORLAT

a) Az 5Ø-es sornak ilyennek kell lennie:
5Ø IF G<11 THEN 3Ø
b) A 3Ø-as sor így néz ki helyesen:
3Ø A$=A$ + "*"

8.3 GYAKORLAT

c) 2Ø-as sor: PRINT (nagy I és nem kis el betű)
A 4Ø-es sor után nincs RETURN
6Ø-as sor: IF X<13 THEN 4Ø (nem pedig X>13)
7Ø-es sor: STOP (és nem STØP)

9. LECKE

9.2. GYAKORLAT

1Ø COLOR 4,5

2Ø COLOR Ø,8

3Ø PRINT " SHIFT és CLR HOME ",

4Ø PRINT" CLR HOME ↓ ← 9-szer → ↓ → ←6-szor→ → ";

5Ø PRINT TI$
6Ø GOTO 4Ø

9.3 GYAKORLAT

5 REM IZLAND ZASZLAJA

1Ø COLOR 4,1

2Ø COLOR Ø,15

2Ø PRINT " SHIFT és CLR HOME ";

3Ø J=1

4Ø PRINT " CTRL és RVS ON ← → 6-szor → → CTRL és WHT ← 1 szóköz → CTRL és RED ← 2 szóköz → CTRL és WHT 1 szóköz"

5Ø J=J+1

6Ø IF J<1Ø THEN 4Ø

7Ø PRINT " CTRL és RVS ON CTRL és WHT ← 7 szóköz → CTRL és RED ← 2 szóköz → CTRL és WHT ← 13 szóköz →";

8Ø J=1

9Ø PRINT " CTRL és RVS ON CTRL és RED 22 szóköz →";

1ØØ J=J+1

```
110 IF J<4 THEN 90

120 PRINT " [CTRL] és [RVS ON] [CTRL] és
      [WHT] ← 7 szóköz → [CTRL] és [RED]
      ← 2 szóköz → [CTRL] és [WHT]
      ← 13 szóköz →";

130 J=1

140 PRINT " [CTRL] és [RVS ON]
[→] ← 6-szor → [→]
      [CTRL] és [WHT] ← 1 szóköz →
      [CTRL] és [RED] ← 2 szóköz →
      [CTRL] és [WHT] ← 1 szóköz →"

150 J=J+1

160 IF J<9 THEN 140

170 PRINT " [CTRL] és [RVS ON]
[→] ← 6-szor → [→]
  [CTRL] és [WHT] ← 1 szóköz →
  [CTRL] és [RED] ← 2 szóköz →
  [CTRL] és [WHT] ← 1 szóköz →";
180 GOTO 180
```

10. LECKE

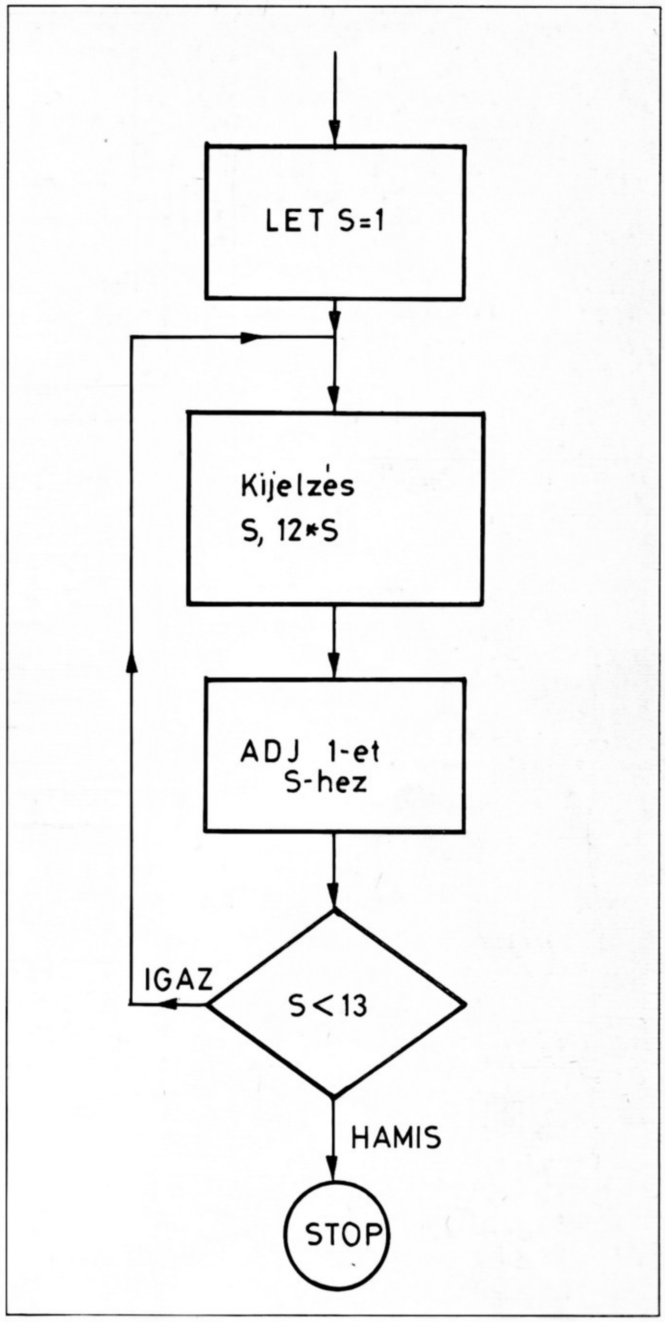
10.2 GYAKORLAT

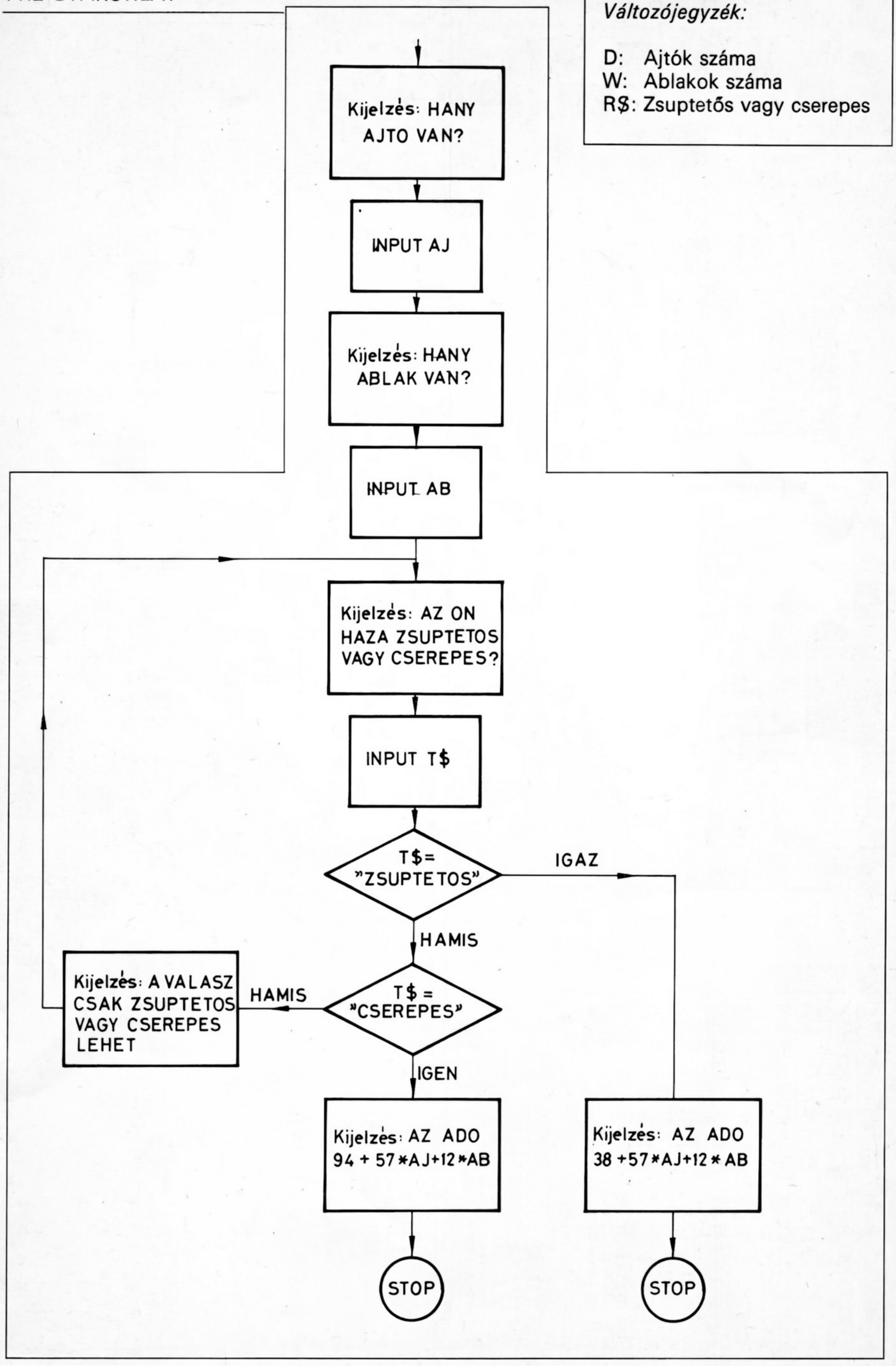
a)
```
10 PRINT "SZORZOTABLA PROGRAM"
20 INPUT "SZORZOSZAM";N
30 X=1
40 PRINT X; "X";N;"AZ";N*X
50 X=X+1
60 IF X<13 THEN 40
70 STOP
```

b)
```
10 PRINT "MI A"
20 INPUT "CSALADI NEVE";S$
30 PRINT "MI A FELESEGE"
40 INPUT "KERESZTNEVE",C$
50 PRINT "A FELESEGE TELJES NEVE"
60 PRINT C$ +" "+S$
70 STOP
```

11. LECKE

11.1 GYAKORLAT




Változójegyzék:
D: Ajtók száma
W: Ablakok száma
R$: Zsuptetős vagy cserepes
Kijelzés: HANY AJTO VAN?
INPUT AJ
Kijelzés: HANY ABLAK VAN?
INPUT AB
Kijelzés: AZ ON HAZA ZSUPTETOS VAGY CSEREPES?
INPUT T$
T$= "ZSUPTETOS"
IGAZ
HAMIS
T$ = "CSEREPES"
HAMIS
Kijelzés: A VALASZ CSAK ZSUPTETOS VAGY CSEREPES LEHET
IGEN
Kijelzés: AZ ADO 94 + 57*AJ+12*AB
Kijelzés: AZ ADO 38 +57*AJ+12*AB
STOP
STOP

```
10 REM RURITANIAI HAZADO
20 PRINT "ADOKISZAMITO
   PROGRAM"
30 INPUT "HANY AJTO VAN";AJ
40 INPUT "HANY ABLAK VAN";AB
50 PRINT "AZ ON HAZA"
60 PRINT "ZSUPTETOS VAGY"
70 INPUT "CSEREPES";T$
80 IF R$ = "ZSUPTETOS" THEN 140
90 IF R$ = "CSEREPES" THEN 160
100 PRINT " A VALASZ CSAK"
110 PRINT "ZSUPTETOS VAGY"
120 PRINT "CSEREPES LEHET"
130 GOTO 50
140 PRINT "AZ ADO";
    38+57*AJ+12*AB
150 STOP
160 PRINT "AZ ADO";
    94+57*AJ+12*AB
170 STOP
```

Helyes válaszok a három variációra:

a) 95 b) 155 c) 364

12. LECKE

12.1 GYAKORLAT

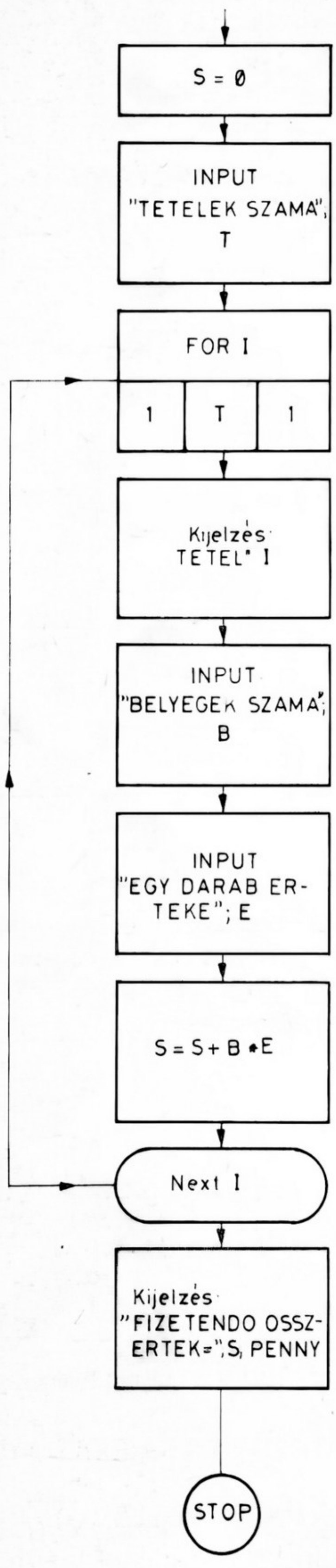
```
10 RS=0                          }  a fordított
20 INPUT "UTESI IDOK SZAMA";J   }  sorrend is helyes
30 FOR Q = 1 TO J
40 INPUT "EREDMENY";S
50 RS=RS+S
60 NEXT Q
70 PRINT "ATLAG=";RS/J
80 STOP
```


12.2 GYAKORLAT

Változójegyzék:

T: Tételek száma
B: Az egy tételhez tartozó bélyegek száma
E: Egy bélyeg értéke egy tételen belül
S: Fizetendő mozgó végösszeg
I: A tételek számozására

```
10 ST=0
20 INPUT "TETELEK SZAMA";T
30 FOR I = 1 TO T
40 PRINT I; "TETEL"
50 INPUT "BELYEGEK SZAMA";B
60 INPUT "EGY DARAB ERTEKE";E
70 S=S+B*E
80 NEXT I
90 PRINT "FIZETENDO
   OSSZERTEK=";S; "PENNY"
100 STOP
```



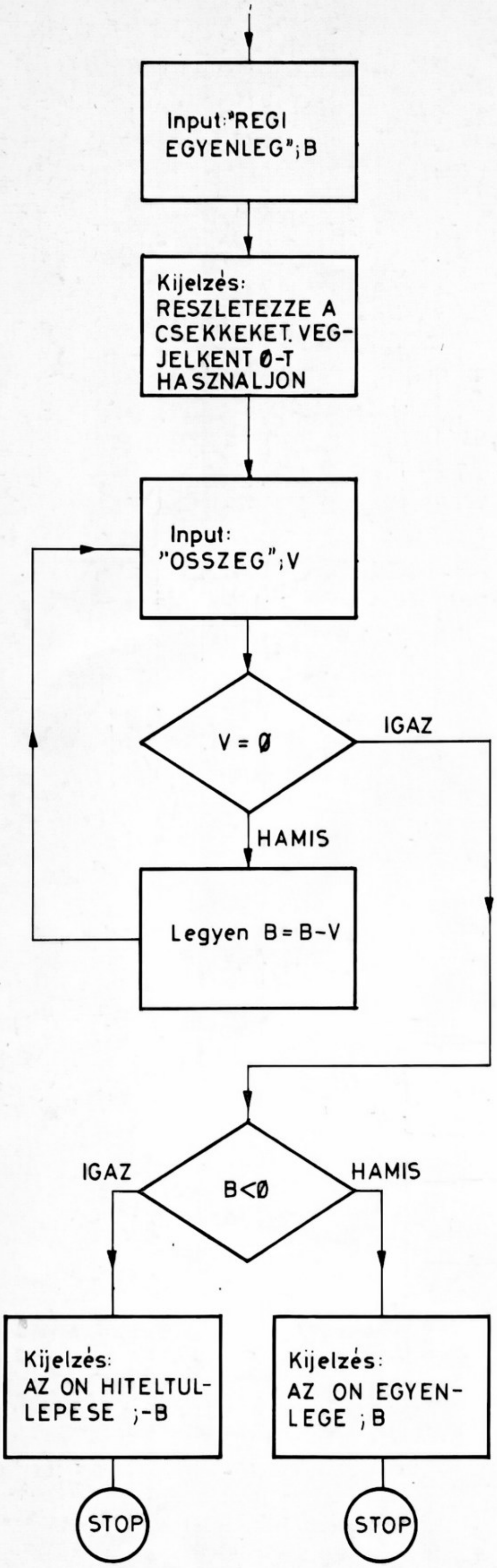
14. LECKE

14.1 GYAKORLAT

Változójegyzék:

B: Jelenlegi egyenleg
V: Minden újabb kifizetés

```
1Ø REM BANK PROGRAM
2Ø INPUT "REGI EGYENLEG";B
3Ø PRINT "RESZLETEZZE
   A CSEKKEKET"
4Ø PRINT "VEGJELKENT Ø-T
   HASZNALJON"
5Ø INPUT "OSSZEG";V
6Ø IF V=Ø THEN 9Ø
7Ø B=B−V
8Ø GOTO 5Ø
9Ø IF B<Ø THEN 12Ø
1ØØ PRINT "AZ ON EGYENLEGE";B;"£"
11Ø STOP
12Ø PRINT "AZ ON HITELTULLEPESE"
13Ø PRINT "−B";"£"
14Ø STOP
```



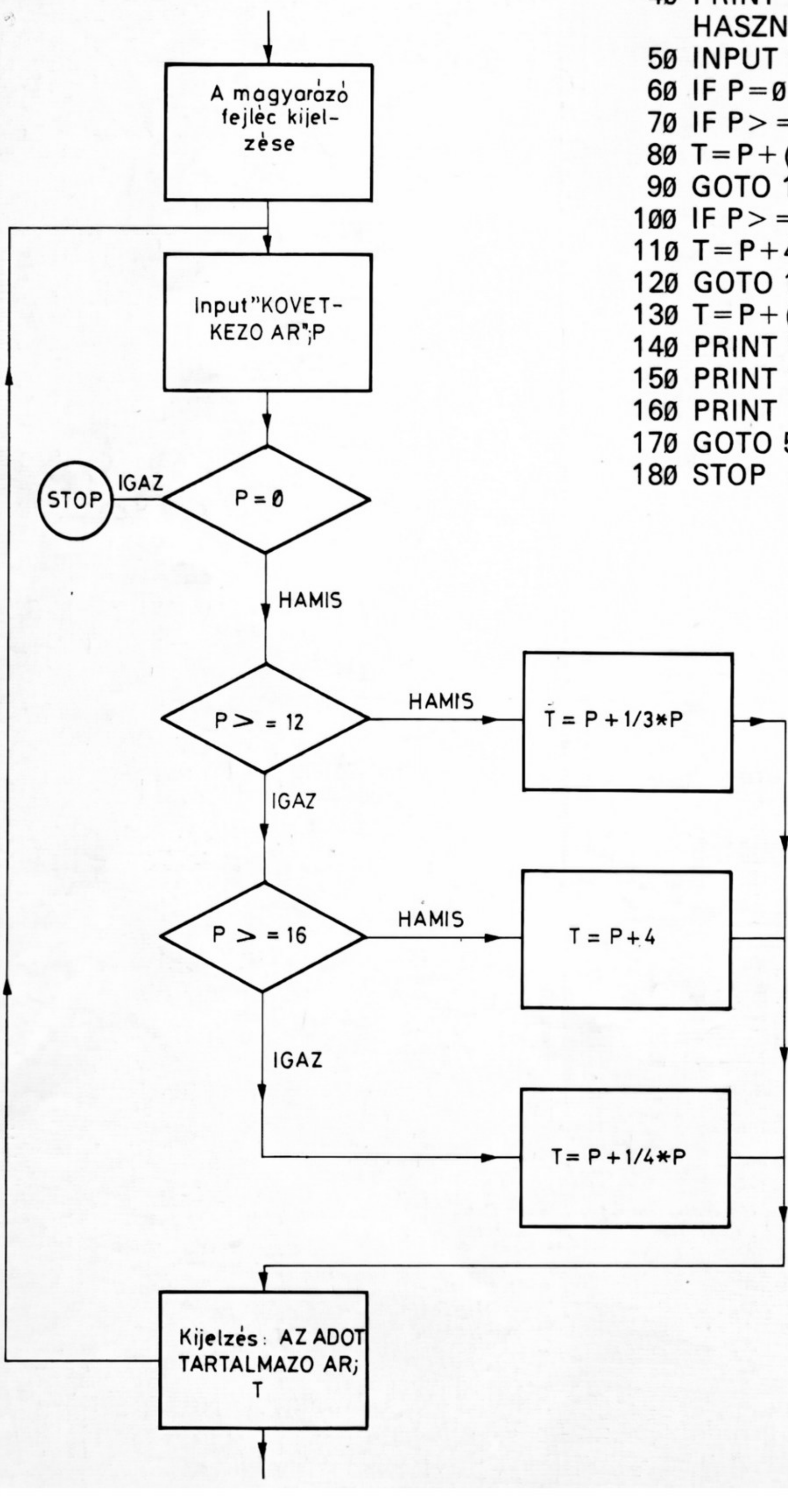
Input:"REGI EGYENLEG";B
Kijelzés: RESZLETEZZE A CSEKKEKET. VEG-JELKENT Ø-T HASZNALJON
Input: "OSSZEG";V
V = Ø
IGAZ
HAMIS
Legyen B=B-V
B<Ø
IGAZ
HAMIS
Kijelzés: AZ ON HITELTUL-LEPESE ;-B
Kijelzés: AZ ON EGYEN-LEGE ;B
STOP
STOP

b)

Változójegyzék:

P: Az óra adómentes ára
T: Az óra adót tartalmazó ára

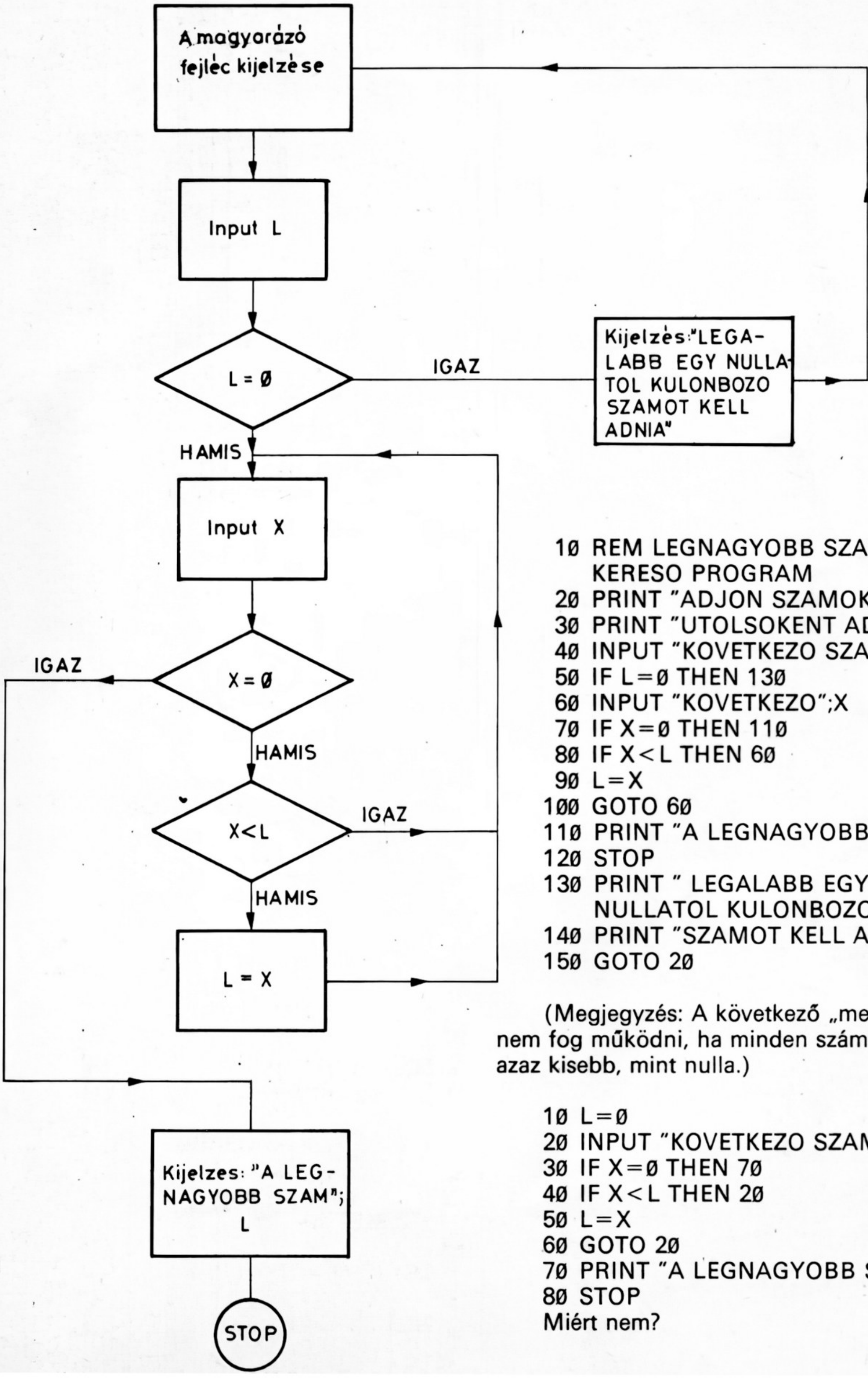
```
 1Ø REM ORA ADO
 2Ø PRINT" SHIFT és CLR HOME ORA
    ADO PROGRAM"
 3Ø PRINT "ADJA MEG AZ
    ADOMENTES ARAKAT"
 4Ø PRINT "VEGJELKENT Ø-T
    HASZNALJON"
 5Ø INPUT "KOVETKEZO AR";P
 6Ø IF P=Ø THEN 18Ø
 7Ø IF P>=12 THEN 1ØØ
 8Ø T=P+(1/3)*P
 9Ø GOTO 14Ø
1ØØ IF P>=16 THEN 13Ø
11Ø T=P+4
12Ø GOTO 14Ø
13Ø T=P+(1/4)*P
14Ø PRINT "AZ ADOT TARTALMAZO"
15Ø PRINT "AR";T
16Ø PRINT
17Ø GOTO 5Ø
18Ø STOP
```



Változójegyzék:

L: Az eddigi legnagyobb szám
X: A következő input szám

c)



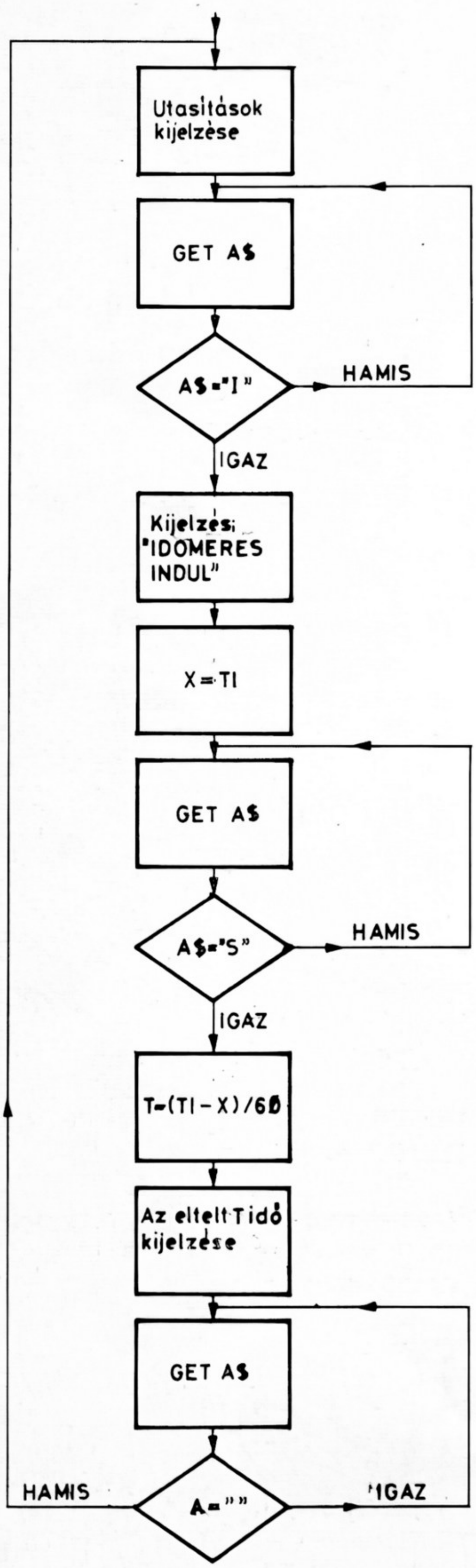

```
 1Ø REM LEGNAGYOBB SZAMOT
    KERESO PROGRAM
 2Ø PRINT "ADJON SZAMOKAT"
 3Ø PRINT "UTOLSOKENT ADJON Ø-T".
 4Ø INPUT "KOVETKEZO SZAM";L
 5Ø IF L=Ø THEN 13Ø
 6Ø INPUT "KOVETKEZO";X
 7Ø IF X=Ø THEN 11Ø
 8Ø IF X<L THEN 6Ø
 9Ø L=X
1ØØ GOTO 6Ø
11Ø PRINT "A LEGNAGYOBB SZAM:";L
12Ø STOP
13Ø PRINT " LEGALABB EGY
    NULLATOL KULONBOZO"
14Ø PRINT "SZAMOT KELL ADNIA"
15Ø GOTO 2Ø
```

(Megjegyzés: A következő „megoldás" nem fog működni, ha minden szám negatív, azaz kisebb, mint nulla.)

```
1Ø L=Ø
2Ø INPUT "KOVETKEZO SZAM";X
3Ø IF X=Ø THEN 7Ø
4Ø IF X<L THEN 2Ø
5Ø L=X
6Ø GOTO 2Ø
7Ø PRINT "A LEGNAGYOBB SZAM:";L
8Ø STOP
```

Miért nem?

15.2 GYAKORLAT (1)

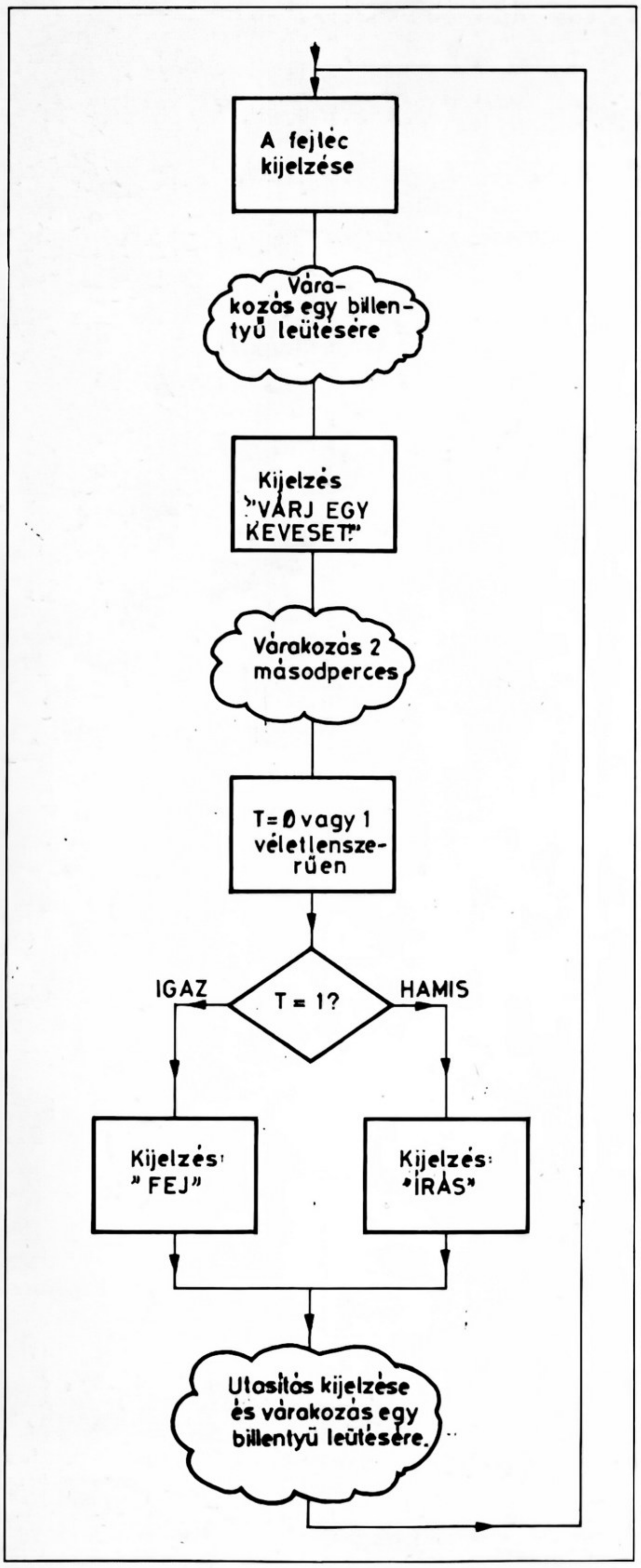


Változójegyzék:

A$: Billentyűzet állapot (GET változója)
X: A belső idő a mérendő időintervallum kezdetekor (hatvanadmásodpercben)
T: Az eltelt idő (másodpercben)

```
  5 REM STOPPERORA
 1Ø PRINT" [SHIFT] és [CLR HOME] "
 2Ø PRINT "STOPPERORA PROGRAM"
 3Ø PRINT
 4Ø PRINT "AHHOZ HOGY
    ELINDULJON A STOPPERORA"
 5Ø PRINT "NYOMJA LE AZ
    I BILLENTYUT"
 6Ø PRINT "HA MEG AKARJA
    ALLITANI NYOMJA LE AZ S-t"
 7Ø GET A$
 8Ø IF A$<>"I" THEN 7Ø
 9Ø X=TI
 95 PRINT "IDOMERES INDUL"
1ØØ GET A$
11Ø IF A$<>"S" THEN 1ØØ
12Ø T = (TI-X)/6Ø
13Ø PRINT "AZ ELTELT IDO"
14Ø PRINT T;"MASODPERC"
15Ø PRINT
16Ø PRINT "NYOMJON LE EGY
    TETSZOLEGES BILLENTYUT"
17Ø PRINT "AZ UJABB
    IDOMERESHEZ"
18Ø GET A$
19Ø IF A$ = ""THEN 18Ø
2ØØ GOTO 1Ø
```

15.2 GYAKORLAT (2)



Változójegyzék:

S$: A billentyűzet állapota (GET változója)
M: A várakozási ciklusban a ciklusváltozó
T: Ø (Fej helyett) vagy 1 (Írás helyett)

```
10 REM EREMDOBAS
20 PRINT " [SHIFT] és [CLR HOME] "
30 PRINT "NYOMJON LE EGY
    BILLENTYUT"
40 PRINT "AZ EREMDOBASHOZ"
50 GET S$
60 IF S$ = ""THEN 50
70 PRINT "VARJ EGY KEVESET"
80 PRINT
90 FOR M = 1 TO 1600
100 NEXT M
110 T=INT(0+2*RND(0))
120 IF T = 1 THEN 150
130 PRINT "IRAS"
140 GOTO 160
150 PRINT "FEJ"
160 PRINT
170 PRINT "NYOMJON LE EGY
    BILLENTYUT"
180 PRINT "A KOVETKEZO
    MENETHEZ"
190 GET S$
200 IF S$ = ""THEN 190
210 GOTO 20
```

15.4 GYAKORLAT

```
5 REM KOCKAJATEK

10 VOL 7

30 PRINT " [SHIFT] és [CLR HOME] [CTRL]
   és [1 BLK] "

40 PRINT "E KOCKAJATEKOT KET
   KOCKAVAL JATSSZAK"

50 PRINT "ELOSZOR FOGADNI
   KELL, MAJD DOBNI"

60 PRINT "HA AZ EREDMENY
   7 VAGY 11, AKKOR NYERT,"

70 PRINT "HA 2-T, 3-AT VAGY 12-T
   DOBOTT, AKKOR"

80 PRINT "VESZTETT. HA BARMI
   MAST DOBOTT, AKKOR"

90 PRINT "EGYBOL NEM NYERHET
   VAGY VESZTHET:"

100 PRINT "ADDIG KELL DOBNIA,
    MIG NEM DOB"

110 PRINT "UGYANANNYIT, MINT
    ELOSZOR"

120 PRINT "(ÉS AKKOR NYERT)"

130 PRINT "VAGY PEDIG 7-ET"

140 PRINT "(ÉS AKKOR VESZTETT)"

150 PRINT

160 PRINT "NYOMJON LE EGY
    BILLENTYUT!"

240 GET A$

250 IF A$ =""THEN 240

255 REM SET A$, B$, C$
    VALTOZOKHOZ A KOCKA
    VONALAI ERTEKADASA

260 A$ = "←9 szóköz→ ┌‾‾‾┐
    ← 5 szóköz→"┌‾‾‾┐

270 B$ = "←9 szóköz→|←7 szóköz
    →|← 5 szóköz→|←7 szóköz→|"

280 C$ = "← 9 szóköz → └───┘
    ← 2 szóköz →└───┘"

285 REM KEZDOTOKE BEALLITASA

290 PRINT " [SHIFT] és [CLR HOME] "

300 INPUT"       KEZDOTOKE";C

305 REM KOVETKEZO FOGADAS
    INDITSA

310 PRINT "A FOGADASHOZ
    NYOMJON LE"

320 PRINT "EGY TETSZOLEGES
    BILLENTYUT"

330 GET R$

340 IF R$=""THEN 330

350 PRINT "       A TOKEJE MOST";C

370 INPUT "       MENNYI A TETJE";W

380 IF W>0 THEN 390

385 PRINT "       NE LEGYEN ENNYIRE
    CSACSKA"

387 GOTO 310

390 IF W<=C THEN 420

400 PRINT "     EZT NEM ENGEDHETI
    MEG MAGANAK"

410 GOTO 310

415 REM AZ ELSO DOBAS
    MEGSZERVEZESE

420 PRINT " [SHIFT] és [CLR HOME] [↓] [↓] [↓]
    ELSO DOBAS (TÉT=";W;")"

430 PRINT [CLR HOME] [↓] 7-szer [↓] ";A$

440 FOR J = 1 TO 5

450 PRINT B$

460 NEXT J

470 PRINT C$
```

```
475 REM 1Ø-59 KULONBOZO
    KOCKAPART MUTAT

48Ø Q=INT(1Ø+5Ø*RND(Ø))

49Ø FOR Z=1 TO Q

5ØØ A=INT(1+6*RND(Ø))

51Ø B=INT(1+6*RND)Ø))

515 REM A-TOL ES B-TOL FUGGO
    HANG

52Ø SOUND 1,7ØØ
    + 3 *(A*A+B*B),4

54Ø PRINT " [CLR HOME] [↓] ← 10-szer → [↓]
    [→] ← 12-szer → [→] "A;
    " [→] ← 11-szer → [→] ";B

56Ø NEXT Z

585 REM UTOLSO A ES B ERTEK
    MARAD

59Ø T=A+B

595 REM UGRAS HA A JATEKOS
    EGYBOL NYERT

6ØØ IF T=7 THEN 1ØØØ/

61Ø IF T=11 THEN 1ØØØ

615 REM UGRAS HA A JATEKOS
    EGYBOL VESZTETT
62Ø IF T=2 THEN 11ØØ

63Ø IF T=3 THEN 11ØØ

64Ø IF T=12 THEN 11ØØ

65Ø PRINT

66Ø PRINT

67Ø PRINT

675 PRINT

68Ø PRINT T;"-T KELL DOBNIA
    7 ELOTT"

7ØØ PRINT " [↓] 8 szor [↓] "NYOMJON
    LE EGY BILLENTYUT"

71Ø GET R$

72Ø IF R$ =""THEN 71Ø

73Ø PRINT " [SHIFT] és [CLR HOME] KOVETKEZO
    DOBAS (TET=";W;")"

74Ø PRINT   ;T;"SZUKSEGES"

75Ø PRINT " [CLR HOME] [↓] ← 4-szer → [↓]"

76Ø PRINT A$

77Ø FOR J=1 TO 5

78Ø PRINT B$

79Ø NEXT J

8ØØ PRINT C$

8Ø5 REM 1Ø-19 KULONBOZO
    KOCKAPART MUTAT

81Ø Q=INT(1Ø+1Ø*REND(Ø))

82Ø FOR Z=1 TO Q

83Ø A=INT(1+6*RND(Ø))

84Ø B=INT(1+6*RND(Ø))

85Ø SOUND1,7ØØ+3*(A*A+B*B),4

87Ø PRINT " [CLR HOME] [↓] ← 8-szor → [↓]
    [→] ← 12-szer → [→] ";A; [→]
    " ← 11-szer → [→] ";B

9ØØ NEXT Z

925 REM IF A+B= T JATEKOS NYER

93Ø IF A+B= T THEN 1ØØØ

935 REM IF A+B=7 JATEKOS
    VESZIT

94Ø IF A+B=7 THEN 11ØØ

945 REM EGYEBKENT UJRA DOB

95Ø GOTO 7ØØ

99Ø REM JATEKOS NYER
```

129

```
1000 PRINT "[↓] ← 7-szer → [↓] NYER"
1005 REM NYEREMENYT A TOKEHEZ
     ADJA
1010 C=C+W
1015 REM DICSOSÉG DIADALENEK
1017 FOR J = 1 TO 500:NEXT J
1020 SOUND 1,834,32
1025 SOUND 1,798,24
1030 SOUND 1,810,8
1035 SOUND 1,834,32
1040 SOUND 1,739,32
1042 SOUND 1,770,8
1044 SOUND 1,798,8
1046 SOUND 1,810,8
1048 SOUND 1,834,8
1050 SOUND 1,810,16
1052 SOUND 1,798,16
1054 SOUND 1,770,64
1095 GOTO 310
1100 REM JATEKOS VESZIT
1110 PRINT "[↓] ← 10-szer → [↓]
     VESZTETT"
1115 REM BOSSZUALLO GYOZELEM
     CSIRIPJE
1120 FOR J = 1 TO 500:NEXT J
1130 FOR X=800 TO 1000 STEP 4
1140 SOUND 1,X,1
1150 SOUND 1,X+23,1
1160 NEXT X
1195 REM VESZTESEG LEVONASA
     A TOKEBOL
1200 C=C-W
1210 IF C>0 THEN 310
1220 PRINT "MOST MEG VAN
     KOPASZTVA"
1230 STOP
```

Az A. Függelék feladatainak megoldásai

1. PÉLDA

```
10 INPUT V
20 INPUT R
30 PRINT "A =";V↑2/R
40 STOP
```

2. PÉLDA

```
10 PRINT "X KEPLET"
20 FOR X = 0 TO 2 STEP 0.2
30 PRINT X; 1/(1+X↑2)
40 NEXT X
50 STOP
```

3. PÉLDA

```
10 PRINT "ADD MEG A HAROM
   OLDALT"
20 INPUT "A";A
30 INPUT "B";B
40 INPUT "C";C
50 S=(A+B+C)/2
60 X=S*(S-A)*(S-B)*(S-C)
70 IF X<0 THEN 100
80 PRINT "A TERULET";SQR(X)
90 STOP
100 PRINT "EZEK NEM LEHETNEK"
110 PRINT „EGY HAROMSZOG
    OLDALAI"
120 STOP
```

Váltójegyzék:

A, B, C: A háromszög három oldala
S: Fél kerület
X: A terület négyzete (ha van egyáltalán)

4. PÉLDA

```
10 REM KICSIT GYORSABB
   VALTOZAT
20 INPUT "LEGMAGASABB ERTEK";L
30 FOR N=3 TO L STEP 2
40 Q=SQR(N)
50 FOR J = 2 TO Q
60 IF N/J=INT(N/J)THEN 90
70 NEXT J
80 PRINT N;
90 NEXT N
100 STOP
```

C. Függelék

HIBAÜZENETEK

Ez a jegyzék azokat a hibákat tartalmazza, amelyek a könyvünkben ismertetett BASIC lehetőségek alkalmazásakor keletkezhetnek. Ennél bonyolultabb programok futtatásakor egyéb hibák is lehetnek.

DIVISION BY ZERO
(Osztás nullával)

A nullával való osztás nem megengedett. Ilyen utasításokban léphet fel a hiba:

    1Ø A=5/Ø vagy

    2Ø B=Q/(J–J)

EXTRA IGNORED
(Felesleg mellőzése)

Ha egy INPUT utasításra válaszolva túl sok tételt (számot vagy füzért) gépelünk be, a gép nem veszi tudomásul a felesleget. A program nem áll le.

ILLEGAL QUANTITY
(Nem megengedett mennyiség)

Egy parancsban vagy utasításban alkalmazott szám túl nagy (vagy túl kicsi). Például a POKE utasítás utáni számnak Ø és 255 között kell lennie. Ez a hiba parancsokban fordulhat elő:

    SOUND 1Ø,5,37 vagy

    COLOR 3ØØ.2

LOAD ERROR
(Betöltési hiba)

A program a mágnesszalagról vagy a mágneslemezről nem helyesen került betöltésre. Ha magnókazettát használ, próbálja megtisztítani az olvasófejet. Az is lehet, hogy eredetileg nem volt jól rögzítve a program, vagy valamilyen mágneses mező tönkretette a szalagot.

NEXT WITHOUT FOR
(NEXT FOR nélkül)

Hibás a program FOR-NEXT szerkezete.

OUT OF MEMORY
(Betelt a tár)

A számítógép tárjában elfogyott az üres hely. Ez csak nagyon hosszú programoknál fordul elő, vagy olyanokkal, amelyek nagy mennyiségű adatot tartalmaznak.

REDO FROM START
(Kezdd elölről)

Ha egy INPUT utasítás egy számra vár, és valami olyasmit írunk be, ami nem szám, a számítógép ezt az üzenetet írja ki, és lehetővé teszi, hogy újra kezdjük.

STRING TOO LONG
(A füzér túl hosszú)

Egy konkatenációval létrejött füzér 255 byte-nál hosszabb

SYNTAX ERROR
(Szintaktikus hiba)

Egy parancs vagy utasítás megszegte a BASIC szintaktikáját (nyelvtani szabályait). Okozhatja a pár nélküli zárójel, a hibásan leírt kulcsszó vagy egy kifejezés helytelen sorrendű elemei.

TYPE MISMATCH
(Típus keveredés)

Ez azt jelenti, hogy számot használtunk füzér helyett vagy fordítva.

VERIFY ERROR
(Ellenőrzésnél hibát talált)

Az ellenőrzési folyamat hibát jelez. Próbálja meg újra kimenteni (SAVE) a programot.

Ára: 226,– Ft